64 PLUS 4





PAŹDZIERNIK 1991 ISSN 0867-3918 INDEKS 377112 CENA 8.000 zł

MIESIĘCZNIK UŻYTKOWNIKÓW KOMPUTERÓW COMMODORE

OGŁOSZENIA

C-64 ze stacją. Ponad 400 programów. Wymienię się oprogramowaniem. Adam Buchta, ul. Mieszka I 12. 69-110 Rzepin, woj. gorzowskie.

Poszukuję literatury na C-64 (Assembler, Pascal, Logo, Forth). T. Jagiełło, ul. 1000-lecia P.P. 21/6, 84-120 Władysławowo.

Sprzedam stację Commodore 1571, Marcin Fijałkowski, Warszawa, tel. 22-68-92, lub 43-94-67.

C-64, magnetofon, joystick, przedłużacze, cartridge, 300 programów, przewód euro-komputer, literatura, bezpłatny transport i montaż w domu. Adam Niklewicz, ul. Karłowicza 2/164, 58-506 Jelenia Góra, tel. 410-39.

Wymienię programy na AMIGĘ. Korusiewicz Przemysław, 43-200 Pszczyna, ul. Chochółka 21.

Tanio sprzedam C-16 z magnetofonem, 2 joysticki, 115 programów. Cena ok. 1.300tys. Marcin Ścisłowski, ul.Dzieci Wrześni 6/15, 41-303 Dąbrowa Górnicza.

Wymienię gry C-64 (taśma). Krzysztof Jagodziński, ul. Sienkiewicza 3/2, 72-420 Dziwnów, tel. 430.

Tanio sprzedam C-64 ze stacją i magnetofonem. Tel. 51-09-98, Jenisz Łukasz, Sierakowskiego 72/58, 91-326 Łódź.

Poszukuję posiadaczy A 500 w celu wymiany doświadczeń. Krzysztof Nowak, ul. Paszkowskiego 6/39, 97-200 Tomaszów Maz.

Sprzedam tanio C 128D, Star NX1000, interface, 2 magnetofony, Final II, literaturę. Miłosz Kłosowicz, ul. Matejki 20/30, 32-510 Jaworzno, tel. 64082.

Sprzedam syntezator mowy do C-64 w zestawie do samodzielnego złożenia, cena 300 tys. Jerzy Andreasik,57-320 Polanica Zdrój, ul. Spółdzielców 10/3.

C-64C, 1541 II, Final III, gwarancja, za 3.6 mln. Grzegorz Terek, Toruń, tel. 48-01-21.

AMIGA-najlepsze gry, programy użytkowe, nowości - wysyłka pocztą. Ekspresowe terminy, katalogi gratis. Dla sklepów rachunki. SOFTSTUDIO, Tysiąclecia 54/6, 31-610 Kraków, tel. (012) 48-51-50.

AMIGA PROGRAMY. Informacja (koperta i znaczek) M. Łajp, ul. Grobla 8/5, 61-858 Poznań.

ODSPRZEDAM TANIO
(po 90tys. zł)
CARTRIDGE X i BLACK BOX
do Commodore C-64.
J. Głowacki
85-870 Bydgoszcz ul. Ogrody 25/179

Grupa THE SPACE PARANOIDS poszukuje nowych członków.
Jeśli jesteś dobrym koderem, muzykiem, grafikiem przyślij próbki swojej pracy (100% odpowiedzi i zwrotu dysku).
Nasz edres: „KALOSZ", ul Benisławskiego 24/c/13, 81-173 GDYNIA.

COMMODORE C64/128,
ATARI 800XL, 65,130XE

Twój komputer zarobi na Ciebie i Twoją rodzinę

3-8mln zł mieslęcznie.

Informacje w Poradniku przesylam za zaliczeniem pocztowym. 27000zł przy odbiorze.
Robert Norton,
39-303 Mielec,
skr. poczt.1

Amiga Public Domain & Shareware Software

Tylko u nas – już od 20 tys. zl za dyskietkę pełną interesującego oprogramowania. Nasza pelna oferta to ponad 500 MB różnorodnego software'u: gry, animacje, edytory tekstu, bazy danych, programy komunikacyjne, antywirusowe i inne użytkowe. W naszych zbiorach m. in. biblioteka Freda Fisha (do nr 470). Na przykład:

Mysz 023		NASA Graphics	- 14 grafik (6 w HAM'ie) o tematyce kosmicznej
Fish 116		Movies	– pakiet pozwalający na tworzenie własnych animacji
Fish 196		HamPics	– 5 super grafik w overscan HAM
Fish 379	m. in.	The A64 Package	– emulator Commodore C64 (wymaga 1 MB RAM)
Fish 413		Aerotoons & Jugette	– 5 animacji (spakowanych – od 0,5 do 1,5 MB RAM)
Fish 418	m. in.	At Movies	– znakomita animacja Erica Schwartza
Fish 456	m. in.	Cheat Sheet	– hasła, rady, instrukcje do ponad 150 gier

i setki innych...

Pełna informacja w katalogu.
Katalog na dysku tylko 15.000 zł.

Nasz adres:



Mysz Shareware
skr. pocztowa 31
81-969 Gdynia 2

Płatność do wyboru: za zaliczeniem pocztowym, czekiem, przekazem
Ceny: 1–5 dyskietek 25.000 zł za sztukę, 6–10 24.000 zł, 11–20 23.000 zł, 21–50 22 000 zł, 51–100 21.000 zł, ponad 100 20 000zł. Powyższa cena obejmuje koszt dyskietki 3,5", kopiowania, opakowania i wysyłki.
W przypadku zaliczenia pocztowego doliczamy opłatę pobieraną przez pocztę za przekaz pieniędzy

---

Klub Komputerowy Stodoła

AMIGA

- oferuje najlepsze stacje dysków 3,5" i 5,25"
- serwis sprzętu firmy Commodore
- literatura (także 64 plus 4)
- akcesoria itp.

Zapraszamy codziennie, oprócz sobót i niedziel w godzinach 11^{00} - 20^{00}

Warszawa ul. Batorego 10
tel. 25-60-31 wew. 35.

Giełdy komputerowe w Stodole, sobota od 10^{00} - 15^{00}

---

64 PLUS 4

miesięcznik nr 10(12) październik 1991
cena 1 egz.: 8000 zł



Wydawca:
ABUK Spółka z o.o.

Redakcja nie ponosi odpowiedzialności za treść ogłoszeń.

Adres redakcji: Redakcja „64 plus 4" 85-166 Bydgoszcz 43 skrytka pocztowa 64
Redagują: Marcin Dudar, Sambor Kuźma, Paweł Sołtysiński, Waldemar Szczygieł (red. nacz.), Krzysztof Kobuz.
Okładka: Piotr Bartz.
Skład: ABUK
Druk: Z.P. POLRASTER 85-353 Bydgoszcz, ul. Orawska 19.

OD REDAKCJI

Wadliwa dystrybucja „64 plus 4 & Amiga” przez przedsiębiorstwo RUCH jest przyczyną kłopotów, jakie mają czytelnicy chcący nabyć nasze pismo. Zdarza się, że otrzymujemy jako zwroty NIETKNIĘTE paczki zbiorcze. Tymczasem czytelnicy sygnalizują, że do wielu kiosków nie dociera ono wcale. Dlatego:

zapraszamy wszystkich chętnych do prowadzenia kolportażu „64 plus 4 & Amiga” (kluby, studia i sklepy komputerowe, księgarnie, osoby indywidualne itd.) do współpracy!

Oferujemy korzystne warunki!

Zainteresowanych prosimy o kontakt z działem dystrybucji pod adresem: Przedsiębiorstwo ABUK, 87-200 Wąbrzeźno, ul. 1 Maja 33.

NOWOŚĆ!

PUBLIC DOMAIN PACK dla COMMODORE 64 NA KASECIE!

Jeszcze w tym roku ukażą się cztery kasety!

Na naszych zestawach znajdują się programy, które do tej pory ukazały się na dyskach PDP, oraz wiele nowości!

Pojedyńcza kaseta kosztuje 30.000zł, wykupienie prenumeraty do końca roku (4 kasety) kosztuje tylko 110tys zł.

Wśród wszystkich, którzy zamówią komplet kaset (blankiet wpłaty zamieszczamy na str. 13) rozlosujemy 10 bezpłatnych prenumerat naszego pisma na rok 1992! Na blankiecie prosimy dopisać PDP-taśma. Spis treści wewnątrz numeru.

Przedsiębiorstwo ABUK S-ka z o.o. oferuje państwu **szybką i tanią obsługę reklamową.** Ogłoszenia drobne od osób indywidualnych (do 10 słów) przyjmujemy bezpłatnie. Większe - 1000 zł za słowo. Reklamy ramkowe (minimalny format - 20 cm^2): 1cm^2 ogłoszenia-**8000zł**, cała strona - **3,0 mln zł**; kolor - odpowiednio 100% drożej.

Ogłoszenia przyjmujemy za pośrednictwem poczty (nasz adres - patrz stopka redakcyjna). Treść ogłoszenia z określeniem formatu reklamy (ewentualnie zamówieniem koloru) prosimy nadsyłać listem poleconym wraz z odcinkiem wpłaty (za pomocą przekazu pieniężnego) na konto Przedsiębiorstwa ABUK Bank Polska Kasa Opieki SA Oddział w Bydgoszczy, konto nr : 5.09011-400522.7-136-11-111.0 Dołączenie do zamówienia odcinka wpłaty przyspieszy zamieszczenie reklamy .

W numerze :

W następnym numerze:

- C-64 - sampling bez samplera!
- AMIGA - język C (c.d.)
- Czy C-128D=128D?

Z daleka i z bliska

- =KB - TALKER= Urządzenie o tak dziwnej nazwie wypuściła na rynek firma Co-Tronics Engineering. Za 69.95$ otrzymujemy przystawkę pozwalająca wykorzystywać na Amidze wszystkie typy klawiatur kompatybilnych z IBM AT. Czego to ludzie nie wymyślą.

- Zwierzyniec powiększa się w szybkim tempie. O tym, że komputery doskonale żyją z pewnym gatunkiem MYSZY wszyscy wiedzą. O doskonałym pożyciu z KOTAMI wie już mniej osób („Kot" to myszka do góry nogami: poruszamy palcami kulkę znajdująca się na nieruchomej podstawce. Czyżby analogie z głaskaniem?). O SZCZURACH słyszeli już nieliczni (to taka myszka „w drugą stronę". Ma toto własny napęd i pisak. Wstarczy postawić takiego szczura na gładkiej powierzchni pokrytej papierem i mamy mini-ploter). „Najgorsze" jest to, że rynek urządzeń I/O tego typu ogromnie się powiększa. Dostępne są mini-tabliczki wrażliwe na dotyk, ergonomicznie wyprofilowane myszy, koty, koto-myszy, nawet klawiatury przypominające odcisk dłoni, obsługiwane jedną dłonią, palce której spoczywają w otworach z przyciskami klawiszy. Czego to ludzie nie wymyślą.

- LabelDex! ($74.95, EasyScript) obiecuje rozwiązać raz na zawsze problemy z organizacją i zarządzaniem listami nazwisk, adresów, telefonów, bibliotekami dyskietek itp. LabelDex! jest kompatybilna z dBASE i współpracuje z ARexxem. Może nie tylko przechowywać dane, lecz np. automatycznie odczytywać informacje o dyskietkach, formatować i drukować tabele itp.

- Podobny do LabelDex! jest mały, szybki i rezydentny Contact 1.2 za $59 z firmy Desktop Utilities. Może być uaktywaniany hot-keyami. Ciekawostką jest fakt, że współpracuje również z drukarkami postscriptowymi.

- ICD przedstawiło wewnętrzne urządzenie do Amigi 500 o nazwie Prima. Tym urządzeniem jest... twardy dysk o pojemności do... 130 MB!!! Nie ma jednak róży bez kolców. Ze względu na konieczności zajęcia miejsca przeznaczonego na wewnętrzną stację dysków dołączany jest Shuffleboard ($25), który pozwala każdą zewnętrzną stację dysków traktować jako DF0:. Wewnętrzne dyski twarde do A500 firmy Novia o pojemnościach 30 i 60 MB nie wymagają usunięcia stacji dysków z A500.

- Znany już u nas chyba od lat Directory Opus został zaprezentowany przez firmę INOVAtronics w cenie $59.

- Król Desktop Publishing na Amidze, Pagestream, doczekał się wersji 2.2. Miejmy nadzieję, że teraz będzie to kawałek naprawdę przemyślanego software'u o potężnych możliwościach.

- O karcie FUSION-FORTY pisałem poprzednim razem. Teraz bliższe dane: Motorola MC68040, 25 MHz, 18-25 MIPS, 3.5-8.0 MFLOPS, 32-bit RAM. To potrafi każdego, kto ma coś wspólnego z komputerami, przyprawić o zawał lub palpitacje serca. Porównajmy dane w tabeli poniżej.

Jarosław Chrostowski

	FUSION-FORTY	COM-MODORE 2630	GVP A3001	IBM i486
Procesor	MC 68040	MC 68030	MC 68030	i80486
Zegar	25 Mhz	25 MHz	25 MHz	25 MHz
MIPS	20+	5.8+	6.4+	15
MFLOPS	3.5+	mniej niż1	mniej niż1	1
Cache	4KB x 2	256 B x 2	256 B x 2	8 KB
Burst	tak	nie	tak	tak
Pamieć(32bit)	4 M standard	2 M standard		

Uczymy się programować

Pamięć komputera można sobie wyobrazić, jako długą ulicę, której domy (bajty) ponumerowane są od 0 do 65535. Każdy dom ma osiem okien (bitów), które mogą być jasne (1), lub ciemne (0), a dzięki którym możemy odczytać jego zawartość. Przy pomocy 8 bitów można przedstawić liczby od 0 (wszystkie bity równe 0) do 255 (wszystkie bity równe 1), inne kombinacje bitów leżą pomiędzy tymi dwoma wartościami.

Owa ulica pamięci posiada swoją książkę adresową o 256 stronach, z których każda może pomieścić 256 adresów. Stanowi ona tablicę zajętości pamięci, której to mocno okrojony opis znajduje się na 228 stronie oryginalnej instrukcji obsługi komputera.

Książka adresowa z 256 adresami.

Na zerowej stronie tablicy pamięci (zeropage) znajdują się wszystkie te adresy, które można reprezentować w postaci pojedynczego bajtu. Większe adresy, trzeba dzielić na młodszy bajt L (low) i starszy bajt H (high).

H = INT(AD/256), L = AD - H*256

W ten sposób H stanowi nr strony, a L nr pozycji na stronie. Dla adresu początkowego pamięci ekranu AD= 3072 mamy np.:

H% = 3072/256 = 12 i L = 3072 - 12*256 = 0

A więc pamięć obrazu rozpoczyna się w lewym górnym rogu strony 12. Jako że dla obliczenia wyższego bajtu wybrano zmienną całkowitą, możemy zrezygnować z funkcji „INT()".

Wskazówka nr 1: funkcję całkowitą można czasami pominąć, stosując zmienną całkowitą.

Strona zerowa stanowi pewnego rodzaju spis treści książki adresowej. Zawiera ona między innymi szereg „wskaźników", tzn. par komórek pamięci leżących koło siebie, które zawierają młodszy i starszy bajt adresu początkowego różnych obszarów pamięci. Jako że w instrukcji obsługi do C-16 i PLUS/4 nie zamieszczono opisu strony zerowej, na rysunku nr 1 przedstawiono wskaźniki, które dzielą pamięć użytkownika na 5 obszarów.

Pierwszy wskaźnik wskazuje na adres 4097 (strona 16) - a jest to obszar programów BASIC'a. Po nim następują zmienne proste oraz łańcuchowe (ARRAY). Tekst zmiennych typu STRING umieszczony jest od końca pamięci w kierunku malejących „numerów domów" po to by przy rozrastających się programach można było skorzystać z dwustronicowego obszaru wolnej pamięci. Gdy i ten zostanie zużyty, pojawia się dobrze znany użytkownikom C-16 komunikat „OUT OF MEMORY".

ADRES L/H	Znaczenie
43/44	Wskaźnik na początek obszaru BASIC'a
45/46	Wskaźnik na początek obszaru zmiennych prostych
47/48	Wskaźnik na początek obszaru zmiennych łańcuchowych
49/50	Wskaźnik na koniec obszaru łańcuchów(+1)
51/52	Wskaźnik na początek obszaru pamięci łańcuchów
55/56	Wskaźnik na granicę pamięci

Rys. 1. Tabela wskaźników strony zerowej C-16 i PLUS/4

W przeciwieństwie do C-64 pamięć dostępna dla programów i zmiennych u jego mniejszych braci nie sięga aż do tzw. granicy RAM'u. C-16 rezerwuje sobie 10 ostatnich bajtów na notatki, a PLUS/4 używa ostatnich 3 stron (768 bajtów) na operacje wejścia / wyjścia.

Tylko początek programów BASIC'a i granica RAM'u mają stałe adresy w procesie tworzenia programów, natomiast wszystkie pozostałe granice ulegają przesuwaniu wraz z każdą nową linijką programu oraz nową definicją zmiennych. Ich stan aktualny można stwierdzić poprzez podstawienie zawartości odpowiednich wskaźników ze strony zerowej do wzoru:

AD = PEEK(L) + 256*PEEK(H).

Ponieważ z tego wzoru korzysta się dosyć często, warto przypisać go klawiszowi funkcyjnemu nr 8:

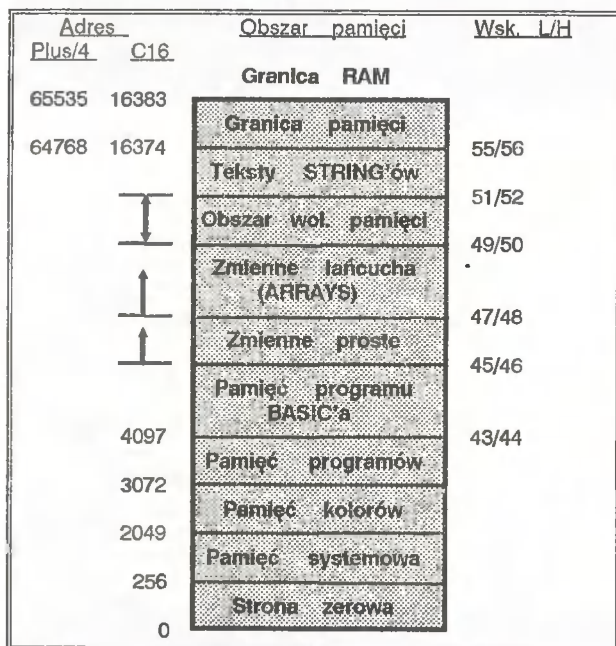
KEY8, „PRINT PEEK(43)+256*PEEK(44)".

Wskazówka nr 2: korzystanie z klawiszy funkcyjnych oszczędza czas.

Po komendzie NEW, przyciśnięcie klawisza funkcyjnego nr 8 i klawisza RETURN spowoduje pojawienie się na ekranie wyniku 4097. Aby sprawdzić pozostałe wskaźniki trzeba zmienić wartości w nawiasach. Adresy 55 i 56 wskazują w C-16 na granicę pamięci o wartości 16374; w przypadku PLUS/4 jest to 64768. Są to owe 12277, lub też 60671 wolne bajty, o których powiadamia nas komputer po włączeniu zasilania.

Wskaźnik 51/52 ma taką samą wartość, jako że nie wprowadzono jeszcze żadnych zmiennych typu STRING. Wskaźniki początku obszaru zmiennych i początku obszaru tablic także są sobie równe i wynoszą 4099. Gdyby pojawiła się jakaś nowa zmienna np. A=5

początek obszaru zmiennych przesunąłby się o 7 bajtów i przyjąłby wartość 4106.



Rys. 2: mapa pamięci C-16 i PLUS/4 (bez grafiki).

Zmienne łańcuchowe zajmują więcej pamięci niż zmienne proste. Polecenie A(4,4)=5 przesuwa pamięć zmiennych pod adres 4720. Powoduje więc zajęcie 614 bajtów cennej pamięci. Aby lepiej zrozumieć istotę przykładu należałoby użyć funkcji FRE. Z rysunku 2 wynika, że wolna pamięć leży pomiędzy adresami wskazywanymi przez komórki 49/50 i 51/52. Rozkaz FRE(0) odejmuje zawartości wyżej wymienionych wskaźników.

W nawiasach rozkazu FRE może się właściwie znajdować dowolna liczba. Jeśli gdziekolwiek w programie pojawi się zmienna łańcuchowa, to komputer zarezerwuje automatycznie miejsce dla całej tablicy. Jeśli wielkość pola nie została zdefiniowana rozkazem DIM to komputer załaduje zmienną do tablicy o rozmiarze 11x11 elementów. Dla dwuwymiarowego łańcucha jest to 121 elementów. Każda zmienna zmiennoprzecinkowa zajmuje 5 bajtów, co daje sumę 605 bajtów. Reszta zostaje zużyta na zarządzanie tablicy. Na marginesie można dodać, że zmienne całkowite zajmują tylko 2 bajty.

```
DIM A(4):FOR I=0 TO 4:A(I)=I+1:NEXT
```

zajmuje mniej pamięci niż

```
DIM A(5):FOR I=1TO5:A(I)=I:NEXT
```

Wskazówka nr 3: pola zmiennych są pożeraczami pamięci. Tam gdzie to możliwe należy korzystać ze zmiennych typu INTEGER. Tablicować każde pole, nawet gdy ma mniej niż 11 elementów. Wykorzystywać elementy zerowe.

Co się stanie ze stałymi granicami, które wyznaczają pamięć użytkownika, gdy przy pomocy rozkazu GRAPHIC 1 lub GRAPHIC 2 włączymy tryb wysokiej rozdzielczości? Otóż początek obszaru BASIC'a pozostanie bez zmian. Natomiast granica pamięci dla programów i danych obniży się o całe 10230 bajtów do adresu 6144. Obszar wolnej pamięci skurczy się do 2045 bajtów. Podobny efekt uzyskamy po wykonaniu rozkazu FRE(0). Gdy powrócimy do trybu tekstowego poprzez GRAPHIC 0, nie odzyskamy żadnej pamięci. Ostatecznie grafika może pozostać. Dopiero po wykonaniu rozkazu GRAPHIC CLR możemy ponownie rozszerzyć pamięć.

Wskazówka nr 4: przed załadowaniem jakiegoś większego programu (większego niż 7 bloków, 1 blok=256 bajtów) należy mu zapewnić rozkazem GRAPHIC CLR odpowiednią ilość pamięci.

Posiadacze PLUS/4 nie mają takich problemów z pamięcią. Po włączeniu grafiki pozostaje im ciągle 48381 wolnych bajtów. W modelu PLUS/4 miejsce na grafikę ekranową rezerwowane jest w dolnym obszarze pamięci, a początek obszaru BASIC'a przesuwany jest w górę o 12Kb (12288 bajtów). Rysunek 3 pokazuje rozkład pamięci w obu modelach przy wyłączonej i włączonej grafice. W PLUS/4 nie ma nawet klawisza z rozkazem GRAPHIC.

Jako przykład zastosowania informacji o wymienionych adresach podana jest procedura MERGE, przy pomocy której można połączyć dwa programy. Najpierw programista wprowadza drugi program do pamięci. Numery jego linii trzeba teraz tak zmienić, aby były większe od tych z pierwszego programu. Przy pomocy rozkazu RENUMBER można to łatwo wykonać. Po przenumerowaniu linii, program musi zostać ponownie nagrany na nośnik danych z nową nazwą. W przypadku PLUS/4, zanim do pamięci zostanie wgrany pierwszy program trzeba wykonać rozkaz „GRAPHICS CLR".

Wskazówka nr 5: procedura Połącz (MERGE)

1. Przy pomocy rozkazu RENUMBER zmienić numerację linii drugiego programu tak by była ona większa od numeracji programu pierwszego.
2. Zapamiętać zmodyfikowany program na nośniku (taśma lub dyskietka).
3. W przypadku komputera PLUS/4 wykonać rozkaz „GRAPHIC CLR".
4. Wgrać pierwszy program do pamięci.
5. Wskaźnik początku obszaru BASIC'a ustawić na koniec programu pierwszego.
   POKE 43, PEEK (45) - 2:POKE 44, PEEK (46)
6. Wgrać drugi program.
   POKE 43,1:POKE 44,16

W ten sposób programista upewnia się, że program zostanie umieszczony w pamięci od adresu 4097. Jego koniec znajduje się dwa bajty przed początkiem obszaru zmiennych, który jest wskazywany przez komórki 45/46. Teraz zmienimy wartość wskaźnika 43/44 (początek programów w BASIC'u) tak by drugi program rozpoczynał się zaraz za pierwszym. Po nagraniu programu trzeba przywrócić starą zawartość tego wskaźnika. Pod adres 46 trzeba „w POKE'ować" 16 (INT(4097/256)), a pod 45 jedynkę.

Grafika	C 16		Plus/4	
	Włączo - na	Wyłączo - na	Włączo - na	Wyłączo - na
Początek BASIC'a	4097	4097	4097	16385
Granica pamięci	16374	6144	64768	64768
Wolna pamięć	12277	2047	60671	48383

Rys. 3: mapa pamięci z grafiką i bez.

W kroku nr 5 może się czasami pojawić błąd, jeśli wynikiem PEEK(45) będzie liczba mniejsza od 2. Można temu zaradzić dopisując linię z komendą REM do pierwszego programu i powtarzając kroki od 5 do 7.

Umieszczenie zmiennych w pamięci.

Po zbadaniu pamięci przy pomocy wskaźników, warto się dokładniej przyjrzeć programowi i zmiennym. Obszary pamięci można przeglądać przy pomocy rozkazu PEEK lub przy pomocy Tedmon'a. Istnieje jeszcze inna możliwość podglądania pracy komputera.

Można w tym celu przemieścić pamięć obrazu. Osiąga się to poprzez zmianę wyższego bajtu wskaźnika 46 na stronie zerowej. Starszy bajt adresu pamięci ekranu ma wartość 12.

Teraz trzeba wprowadzić następujący rozkaz:

```
POKE 46,12:CLR
```

Uwaga! CLR powoduje, że „komunikat o tej zmianie" dotrze do wszystkich „końcówek" systemu operacyjnego. Przy pomocy klawiszy CBM i SHIFT zostanie włączony tryb małych liter, a ekran ulegnie wyczyszczeniu. Kilka linijek poniżej programista definiuje pierwszą zmienną: ab=0.

Po przyciśnięciu RETURN'a w najwyższej linijce pojawi się siedem znaków. Pierwsze dwa znaki (duże litery) stanowią nazwę zmiennej. Pozycje 3 do 7 zawierają wartość przypisaną zmiennej „ab". W naszym przykładzie jest to zero i dlatego owe pozycje wypełnione są znakami „@". Programiści dość często będą się stykać z tym osobliwym znakiem, któremu w tablicy kodów ekranowych odpowiada wartość 0.

Omówimy teraz temat kodowania liczb. Tabele kodów w podręczniku obsługi (str 213 do 217) nadają się wyśmienicie do wychwycenia różnic pomiędzy kodami ekranowymi oraz ASCII. Wszystkie znaki, które zostają wprowadzane z klawiatury do pamięci są przechowywane w formacie ASCII. Dla małych liter w kodzie ASCII a=65, a b=66. Natomiast każdemu znakowi wyświetlanemu na monitorze odpowiada wartość zgodnie z tabelą kodów ekranowych i tak np. liczbie 65 odpowiada „A" a 66 „B".

Stąd bieże się tajemnicza zamiana małych liter w duże.

Zmienne w grupach po siedem bajtów.

Do czego właściwie potrzebne są dwa systemy kodowania? Kody ekranowe służą tylko do wyprowadzania danych. Połowa 256 możliwych kodów używana jest do oznaczania znaków inwersyjnych. Z drugiej strony potrzebny jest szereg kodów sterujących (np. do sterowania kursorem), których nie można przedstawić graficznie. Ponadto duże litery leżą o 64 wyżej niż małe w tabeli kodów, a odpowiadające im litery w inwersji posiadają kody zwiększone o 128 np.: „a"=1, „A"=65, „a" w inwersji=129 i „A" w inwersji=193.

Pozostaje jeszcze niewiadomym, jak komputer rozróżnia typy zmiennych. Następująca linia wyjaśni tą kwestię:

```
ab%=0:ab=":? fnab
```

Wprawdzie ostatni rozkaz spowoduje błąd, ale nie ma to większego znaczenia. Na ekranie pojawią się w rezultacie cztery siedmio-bajtowe grupy. Wszystkie rozpoczynają się od nazwy zmiennej AB, ale tylko w zmiennej zmiennoprzecinkowej są one normalnie wyświetlane. W zmiennej całkowitej oba znaki są wyświetlane inwersyjnie, w zmiennej typu STRING tylko drugi znak, a w zmiennych funkcyjnych tylko pierwszy. Znaki inwersyjne są bardzo łatwo rozpoznawalne dla komputera albowiem ich najstarszy bit jest równy 1 (bit o wadze 128).

Wszystkie proste zmienne przedstawiane są w grupach siedmiobajtowych, pomimo iż zmiennym całkowitym wystarczają cztery bajty, a łańcuchowym pięć. Zaletą takiej reprezentacji jest to, że przy poszukiwaniu zmiennej wystarczy gdy komputer sprawdzi co siódmy bajt. Zmienne łańcuchowe nie pasują jednak do tego schematu. Z tego też względu wymagają oddzielnego obszaru pamięci, a po każdej nazwie zmiennej, dwóch bajtów, które określają położenie następnego pola zmiennej. W grupie siedmiu bajtów zmiennej typu STRING znajduje się tylko deskryptor łańcucha. Składa się na niego długość łańcucha oraz adres miejsca w pamięci, gdzie znajduje się łańcuch.

Deklaracja zmiennych.

Reguły tworzenia nazw zmiennych:

1. Nazwa powinna się składać z jednej lub większej ilości liter lub cyfr. Pierwszym znakiem musi być litera.
2. Nazwy składające się z jednej litery są przedłużane przez komputer o zero.
3. Tylko pierwsze dwa znaki są znaczące.
4. Nazwy, które zawierają zarezerwowane hasło zostaną odrzucone. Ułożenie w pamięci zawsze odpowiada kolejności wprowadzania. Gdy następuje odwołanie do zmiennej, komputer przeszukuje pamięć zmiennych, poszukując zmiennej o tej samej nazwie. Często używane zmienne powinny być z tego powodu deklarowane na początku programu. Np.:

```
10C=0:A$=„":F%=0
```

Bardziej elegancko jest zastosować w takim miejscu rozkaz DIM, o czym wie niewielu programistów. A więc przy pomocy klawisza CLR czyścimy ekran i wypisujemy:

```
0 DIM 10C,A$,F%
```

Zmienne zostały umieszczone w pamięci z pustymi wartościami. Nawet gdy przypiszemy im nową wartość ich miejsce w pamięci zmiennych pozostanie niezmienione.

Dla częściej używanych zmiennych należy rezerwować miejsce w pamięci. Do tego właśnie służy deklarowanie zmiennych w pierwszej linii programu.

Opracowane na podstawie RUN nr 2/87.

Jak zmienić C-64 z systemu NTSC w PAL.

Do naszego kraju dociera wiele egzemplarzy C-64 pracujących w systemie NTSC. Pochodzą one głównie z USA. Nabywcy tych maszyn najczęściej nie orientują się iż będą mieli w Polsce kłopoty z uzyskaniem prawidłowego obrazu i koloru.

Są dwie przyczyny tych kłopotów:
- mamy odmienne systemy przesyłania i kodowania koloru (w Europie obowiązuje system PAL w USA i Japonii NTSC);
- mamy różne częstotliwości zasilania, z czym związane są częstotliwości odchylanie odbiorników TV.

Przynajmniej liczba linii ekranu (625 w Europie i 545 w Japonii i USA) nie ma w tym przypadku znaczenia. Jeśli wystarczy obraz czarno - biały, to istnieje możliwość dostrojenia starego odbiornika bez otwierania komputera. Tutaj jednak tym się nie będziemy zajmować, a zajmiemy się C - 64.

Potrzeba kilku elementów aby zmienić tryb pracy z NTSC w PAL. Obawiam się, że może to być raczej kosztowne, jednak będzie to lepsze niż nie używanie komputera. Niektóre z elementów można kupić w sklepie RTV, ale większość z nich tylko na giełdach lub w punktach serwisowych.

Oto co jest potrzebne:
> nowy zasilacz - nie próbój podłączać amerykańskiego zasilacza do gniazdka (można także zastosować transformator obniżający napięcie wejściowe 220V do napięcia wyjściowego 117V i wykorzystywać oryginalny zasilacz),
> 40 - nóżkowy układ scalony oznaczony numerem 6569 (VIC),
> kwarc na częstotliwość 17.734472 MHz,
> miniaturowy kondensator ceramiczny 15 pF.

Aby dokonać przeróbki potrzebne jest doświadczenie w lutowaniu. Trzeba także powiedzieć, że jeśli otworzysz komputer, to utracisz gwarancję, co może być jednak nieistotne, ponieważ gwarancja jest tylko ważna w kraju zakupu.

Wyłącz komputer i wyciągnij wtyczkę zasilania. Otwórz obudowę, odkręcając trzy śruby w dolnej pokrywie, odegnij klawiaturę do góry, zwróć uwagę, aby nie uszkodzić przewodów do wskaźnika LED. Jeśli chcesz, możesz je , jak i wiązkę przewodów klawiatury, odłączyć. Oderwij foliową tekturę od ekranu portu cartridge'a. Przegnij ją do siebie na krawędzi.

Zlokalizuj układ scalony oznaczony U19. W starszych wersjach komputera układ ten znajdował się w blaszanym ekranie z przykrywką. Jeśli tak to musisz ostrożnie usunąć ją, używając małego wkrętaka. Układ U19 ma 40 nóżek i powinien być oznaczony „6567". Wyjmij go z podstawki (możesz użyć dwa małe wkrętaki i podważyć układ z dwóch stron).

Zapamiętaj kierunek układu, zwykle oznaczony małym wycięciem w jednym z krótszych boków.

Teraz włóż w tym samym kierunku nowy układ „6569". Może zaistnieć potrzeba wygięcia nóżek nieco do środka, aby pasowały w podstawkę.

Upewnij się, czy wszystkie nóżki są dokładnie ponad otworami, a następnie naciśnij cały układ, tak aby wszystkie nóżki weszły jednocześnie. W tej samej strefie płyty zlokalizuj C70. Wylutuj stary i wlutuj nowy kondensator. Obok niego znajdziesz kwarc. Także go wymień. Teraz pozostała jeszcze jedna rzecz do przelutowania. Jest to łączówka. W przypadku NTSC łączówka jest między literami E1 i E3 wydrukowanymi na płytce. Wylutuj łączówkę i przesuń do sąsiedniej pozycji pomiędzy oznaczeniami PAL i E2.

Teraz usuwamy pokrywkę modulatora . W większości komputerów, pokrywka była przylutowana w jednym punkcie. Aby ją usunąć, należy jedną ręką odlutowywać, a drugą oderwać pokrywkę. Następnie sprawdzamy, czy nie ma żadnych pozostawionych przewodów i innych rzeczy, które mogłyby spowodować zwarcie na płytce montażowej (nie jest konieczne podłączenie klawiatury). Teraz podłącz zasilanie i włącz komputer. Powinieneś zobaczyć coś, co może się stać kolorowym obrazem. Zlokalizuj potencjometr regulacyjny R27, w pobliżu kwarcu, obracaj go delikatnie. Powinieneś uzyskać stabilny kolorowy obraz.

Jeśli jesteś jeszcze niezadowolony, musisz przejść do modulatora. Wewnątrz jego możesz zobaczyć kilka punktów regulacyjnych: wyglądają inaczej, ale każdy z nich ma szczelinę na wkrętak. Ta regulacja powinna być wykonana za pomocą bardzo małego śrubokręta wykonanego z niemagnetycznego materiału, np. plastiku lub miedzi. Należy przeprowadzić regulację wszystkich punktów, obracając je w obie strony od pozycji oryginalnej i obserwując obraz. Powinno znaleźć się pozycje, w których obraz jest optymalny. Może okazać się, że poprzednie regulacje powinny być powtórzone, ponieważ są one zależne jedna od drugiej. Ostatecznie powinniśmy otrzymać dobry obraz. Rób to systematycznie i zawsze pamiętaj ile obrotów wykonałeś od pozycji wyjściowej.

Teraz wyłącz komputer, podłącz klawiaturę załaduj program z dźwiękiem lub wykonaj kilka deklaracji POKE, aby wytworzyć jakiś dźwięk.

Jeśli nie ma dźwięku lub jest zniekształcony, to czas spróbować zestroić te punkty w modulatorze, które wydadzą się nie mieć żadnego wpłwu na optymalizację obrazu. Wykonując to powinieneś uzyskać dźwięk wolny od zniekształceń. Jeśli te regulacje pogorszyły znowu obraz, należy wykonać wszystkie regulacje związane z jakością obrazu raz jeszcze, a następnie regulację dźwięku, aż uzyskasz zadawalające wyniki. Na koniec złóż komputer w kolejności odwrotnej niż przy rozbiórce.

Powodzenia !

Opr. H.S.

JAK UKRYĆ ?



Często się zdarza, że pisząc programy w języku BASIC chwalimy sobie tą jego zaletę, iż w stosunkowo nieskomplikowany sposób dokonać można niewielkich poprawek w treści programu, odnaleźć błąd czy też zmienić teksty. Każdy kij ma jednak dwa końce - tak napisany program nie posiada praktycznie żadnych zabezpieczeń przed niepożądaną ingerencją nieproszonych „programistów". Z praktyki wiadomo, że odpowiednie POKE'i potrafią porządnie utrudnić korzystanie z instrukcji LIST.

Cóż jednak z tego - odpowiednie wartości należy wpisać i najczęściej jest to realizowane przez uruchomiony program. Nietrudno jednak dojść do wniosku, że nikt nie będzie próbował w tym programie grzebać po jego uruchomieniu - zaraz po wgraniu go do pamięci komputera mamy go „na patelni" i możemy z nim zrobić, co tylko nam przyjdzie do głowy. Również sposób polegający na wpisaniu pierwszej linii programu w następujący sposób nie chroni programu:

```
10 REM (Shift+L)
```

Program co prawda nie daje się wylistować komendą LIST (pojawia się komunikat Syntax Error), ale użycie LIST 11 - rozwiązuje sprawę. Istnieje jednak lepsze zabezpieczenie, które chciałbym Czytelnikom przedstawić i polecić do wykorzystania.

Jest ono o wiele skuteczniejsze i pozwala na wybiórcze „blokowanie" linii, których ktoś niepowołany nie powinien tak łatwo zobaczyć. Aby skorzystać z tego zabezpieczenia, konieczne będzie posłużenie się prostą procedurą napisaną w języku maszynowym. Przy założeniu, że zamieszczona tutaj procedura znajduje się w pamięci, możemy rozpocząć zabezpieczanie. Polega to na wytypowaniu wskazanych linii przez umieszczenie pięciu znaków dwukropka po numerze każdej z nich, co nie ma wpływu na działanie rozkazów umieszczonych za nimi. Np.:

```
10:::::PRINT "Ta linia bedzie ukryta"

20:::::X=4:B=12:C=B/X
```

Po wywołaniu naszej procedury (SYS 50000) po użyciu komendy LIST uzyskamy następujący efekt:

```
10

20
```

...czyli wyświetlone zostaną tylko numery linii, co nie będzie jednak przeszkadzać w realizacji programu.

Dla zainteresowanych postanowiłem podać wspomnianą wyżej procedurę w dwóch formach: tekstu źródłowego dla TurboAssemblera, oraz jako loader w języku BASIC (do bezpośredniego wykorzystania).

Tekst (tzw. source - źródło) dla assemblera:

```
        *=50000

        LDA   $2B
        STA   $FB
        LDA   $2C
        STA   $FC
LOOP    LDY   #$08
        LDX   #$04
LOOP    0LDA  ($FB),Y
        CMP   #$3A
        BNE   LOOP1
        DEX
        BMI   LOOP2
        DEY
        BNE   LOOP0
LOOP2   LDA   #$00
        STA   ($FB),Y
LOOP1   LDY   #$00
        LDA   ($FB),Y
        TAX
        INY
        LDA   ($FB),Y
        STA   $FC
        STX   $FB
        CMP   $2E
        BCC   LOOP
        CPX   $2D
        BCC   LOOP
        RTS
```

Loader w j. BASIC:

```
10  FOR T=50000 TO 50048: READ C:POKE T,C:
    NEXT:PRINT "URUCHOMIENIE: SYS50000"
99  :
100 DATA165,43,133,251,165,44,133,252,160,8,
    162,4,177,251,201,58,208,10,202,48
110 DATA3,136,208,244,169,0,145,251,160,0,177,
    251,170,200,177,251,133,252,134
120 DATA251,197,46,144,220,228,45,144,216,96
```

Paweł Sołtysiński

JAK ZROBIĆ DEMO (7)

W poprzednich odcinkach naszego cyklu przedstawiliśmy zasady używania graficznego trybu przerwań, pisania procedur scroll'i i tym podobnych zagadnień. Należy jednak pamiętać, że sztuka tworzenia dem polega na właściym łączeniu poszczególnych procedur w jedną, spójną całość.

Dziś chciałbym przedstawić Czytelnikom prosty program demonstracyjny, w którym zostały połączone takie elementy, jak animowane kolorowe paski (tzw. colour bars) i scroll. Daje on bardzo przyjemne efekty wizualne i jest wdzięcznym materiałem do analizy. Całość należy wpisać pod dowolny assembler (najwyżej z niewielkimi zmianami), w artykule użyto Turbo Assemblera.

```
         *=$1000  ;adres początku assemblacji

timetab  =$0e00  ;położenie tabeli opóźnień czasowych
colortab =$0f00  ;położenie tablicy wyśw. kolorów
scrline  =$05b8  ;adres początku lini dla scroll'a


         jsr  $e544          ;kasowanie ekranu
         ldx  #$27
         lda  #$01
loop1    sta  scrline+$d400,x ;ustawianie atrybutów
         dex
         bpl  loop1
         inx
         txa
loop2    sta  colortab,x     ;czyszczenie tablicy kolorów
         inx
         bne  loop2
loop6    ldy  #$07
loop5    cpy  #$00
         bne  loop3
         lda  #$01
         bne  loop7
loop3    lda  #$08
loop7    sta  timetab,x      ;ustawianie opóźnień
                             ;czasowych
         inx
         beq  loop4          ;koniec ustawiania
         dey
         bpl  loop5
         bmi  loop6
loop4    inx
         stx  colortab       ;pierwsza linia biała
         stx  colortab+$72   ;ostatnia też
         jsr  scrlinit       ;ustawianie parametrów scroll'a
         sei                 ;procedura ustawiania
                             ;warunków IRQ
         lda  #$7f
         sta  $dc0d
         ldx  #$00
         stx  $dc0e
         inx
         stx  $d01a
         lda  #$1b
         sta  $d011
         lda  #$53
         sta  $d012
         lda  #q
         ldx  #irq
         sta  $0314
         stx  $0315          ;ustawianie wektora IRQ
         cli
keytest  jsr  $ffe4          ;testowanie klawiatury
         cmp  #$20           ;czy to spacja
         bne  keytest        ;nie testuj dalej
         sei
         jsr  $fda3
         jsr  $fd15
         jmp  $e518          ;powrót do Basic
irq      ldx  #$00
ldy      time tab,x
         irq  dey
         bne  irq1
         lda  colortab,x
         sta  $d020
         sta  $d021
         inx
         cpx  #$74
         bne  irq+2
         jsr  scroll         ;wywołanie scroll'a z IRQ
         lda  $22
         sta  $d016
         jsr  roll           ;przewiń tablicę kolorów wzorca
         jsr  setcol         ;wstaw kolory
         inc  $d019          ;potwierdź następne przerwanie
         jmp  $ea31          ;skok do ROM - IRQ
scrlinit lda  #$07
         sta  $22
init     lda  #              ;ustawianie adresu tekstu
         ldx  #txt
         sta  $20
         stx  $21
         rts
scroll   dec  $22
         bmi  scr2
         rts
scr2     lda  #$7
         sta  $22
         ldx  #$00
scr3     lda  scrline+1,x    ;przesuniecie tekstu na ekranie
         sta  scrline,x      ;o jeden znak w lewo
         inx
         cpx  #$27
         bne  scr3
         ldy  #$00
scr0     lda  ($20),y        ;odczyt kolejnej litery
         and  #$3f           ;konwersja: ASCII=SCREEN
                             ;CODE
         bne  scr4           ;nie ZERO - nie koniec tekstu
         jsr  init           ;ustawienie ponownie wektora
                             ;tekstu
         bne  scr0           ;odczytaj znak ponownie
scr4     sta  scrline+$27
         inc  $20
         bne  scr5
```

```
        inc   $21
        scr   5rts
roll    lda   #$00
        beq   rl0
        dec   roll+1
        rts
rl0     lda   #$03
        sta   roll+1          ;ustawienie petli opóźniającej
        ldx   #$00
        lda   color
        pha
rl1     lda   color+1,x
        sta   color,x
        inx
        cpx   #$0d
        bne   rl1
        pla
        sta   color+13
        rts
color   byte   9,2,8,10,15,7,1
        byte   7,15,10,8,2,9,0

        set   colldx  #$00
        lda   color,x         ;wystawienie tablicy kolorów
        sta   colortab+1,x
        lda   color+1,x
        sta   colortab+9,x
        lda   color+2,x
        sta
        lda   color+3,x
        sta   colortab+25,x
        lda   color+4,x
        sta   colortab+33,x
        lda   color+5,x
        sta   colortab+41,x
        inx
        cpx   #$08
        bne   setcol+2
        ldx   #$01
        ldy   #$71
loop10  lda   colortab,x
        sta   colortab,y
        inx
        dey
        cpy   #$40
        bne   loop10
        rts
```

Paweł Sołtysiński

UWAGA UŻYTKOWNICY PROGRAMU VOICETRACKER V 4.0!

Niektórzy z Was sygnalizują, że nie wiedzą jak uruchomić demonstracje muzyczne znajdujące się na dyskietce lub taśmie. Otóż demonstracje te należy wgrywać nie jako samodzielne programy, ale jako moduły programu. Po wgraniu Voicetracker'a i wejściu do głownego okna edycji wybieramy opcję taśma lub dysk, a następnie wprowadzamy komendę LOAD (klawisz „1"). Komputer zapyta o tytuł pliku jaki chcemy wprowadzić (wpisujemy np. TUNE 01), a następnie wczyta go do pamięci. Po przejściu do głównego menu klawiszem F1 uruchamiamy daną demonstrację.

REDAKCJA

Programy zwariowane, ciekawe i takie sobie.

Przedstawiamy dzisiaj kolejne parę programów z bujnej łączki programów opisanych w tytule, których jedną ze wspólnych cech jest to, że są przede wszystkim krótkie.

A oto pierwszy z nich: po uruchomieniu go, wszystkie procedury drukowania znaków na ekranie (PRINT) zostają zwolnione, a drukowi każdego znaku towarzyszy cichutkie „piknięcie", co daje „komputerowy" efekt, wymarzony przy wszelkiej maści programach naszej produkcji:

```
10 FOR T=272 TO 307:READ C:POKE T,C:NEXT:SYS
   272
20 DATA169,27,141,38,3,169,1,141,39,3,96,72,173,17,
   208,16,251,173,17
30 DATA208,48,251,169,15,141,24,212,169,0,141,24,
   212,104,76,202,241
```

Następnym będzie bardzo prosty programik-kalejdoskop, może niezbyt ładny, ale za to na pewno prosty:

```
5   K=54272:POKE 53280,0:POKE 53281,0
10  PRINT CHR$(147)
20  X1=INT(RND(10)*20):Y1=INT(RND(10)*12):C=INT
    (RND(10)*16)AND 7
30  X=X1:Y=Y1:GOSUB 100
40  X=39-X1:GOSUB 100
50  Y=23-Y1:X=X1:GOSUB 100
60  X=39-X1:GOSUB 100:GET A$:IF A$=" " THEN 10
70  GOTO 20
100 AD=1024+Y*40+X:POKE AD,224:AD=AD+K:
    POKE AD,C:RETURN
```

Aby skasować ekran podczas pracy kalejdoskopu, wystarczy nacisnąć klawisz spacji. A teraz coś dla poszukujących procedury odczytu katalogu dyskietki z poziomu BASIC:

```
10 OPEN1,8,0,"$":POKE 781,1;SYS 65478:GET A$,A$
20 GET A$,A$:IF ST=64 THEN SYS 65484:CLOSE1:END
30 GETA$,B$:PRINTCHR$(157);ASC(A$+CHR$(0))+
   256*ASC(B$+CHR$(0));
40 GET A$:PRINT A$;:IF A$<>" " THEN 40
50 PRINT:GOTO 20
```

Powyższą procedurę można bez większego kłopotu zastosować we własnym programie, pod warunkiem odpowiedniego przenumerowania linii i ewentualnego użycia rozkazu RETURN, o ile planujemy wykorzystać ją jako podprogram.

Polonus

AUTO - BOOT dla C - 64

Programy typu BOOT są bardzo wygodną formą samouruchomiania programów ładowanych z dyskietki. Pomijając konieczność wpisywania przez użytkownika rozkazów typu RUN czy SYS po wgraniu żądanego programu, tzw. booter jest bardzo dobrym początkiem jakiegoś zabezpieczenia (o ile ktoś planuje zabezpieczanie). Ich zasada działania jest bardzo prosta: osoba, która chce wgrać dany program, wgrywa jego pierwszy element (najczęściej właśnie pod nazwą BOOT) z zastrzeżeniem jego nierelokowalności (tzn. LOAD "BOOT",8,1). Tak wgrany program sam się uruchamia (w naszym przykładzie będziemy korzystać z obszaru zarezerwowanego dla stosu procesora), a następnie ładuje i uruchamia program o zaprogramowanej wcześniej nazwie (u nas będzie możliwe nadanie dwuliterowej nazwy).

Po uruchomieniu wydrukowanego poniżej listingu nastąpi załadowanie kodu z wierszy DATA, pytanie o rodzaj programu, który przez BOOT będzie wgrywany (BASIC czy kod maszynowy) oraz ewentualne pytanie o adres startu, o ile ładowany program nie został napisany w języku BASIC.

```
10   AD=49152
20   READ A$:IF A$="XX" THEN 70
30   GOSUB 40:POKE AD,A:AD=AD+1:GOTO 20
40   B$=LEFT$(A$,1):GOSUB 60:A=C*16
50   B$=RIGHT$(A$,1):GOSUB 60:A=A+C:RETURN
60   IF ASC(B$)>64 THEN C=ASC(B$)-55:RETURN
65   C=ASC(B$)-48:RETURN
70   PRINT "[B]ASIC CZY [K]OD MASZYNOWY?"
80   GET A$:IF A$<>"K" AND A$<> "B" THEN 80
84   IF A$="B" THEN 90
85   INPUT "PODAJ ADRES STARTU:";AA
86   POKE 49260,76:C=INT(AA/256):D=AA-256*C
87   POKE 49261,D:POKE 49262,C
90   PRINT "PODAJ NAZWE LADOWANEGO ZBIORU".
91   INPUT "(2 ZNAKI):";N$:IF LEN(N$)<>2 THEN 91
92   POKE 49266,ASC(LEFT$(N$,1)):POKE 49267,
     ASC(RIGHT$(N$,1))
93   SYS 49152
94   :
1000 DATA A9,01,A8,A2,08,20,BA,FF
1008 DATA A9,04,A2,3C,A0,C0,20,BD
1010 DATA FF,20,C0,FF,A2,01,20,C9
1018 DATA FF,A9,01,20,D2,FF,20,D2
1020 DATA FF,A0,00,99,74,C0,C8,D0
1028 DATA FA,B9,40,C0,20,D2,FF,C8
1030 DATA C0,FF,D0,F5,A9,01,20,C3
1038 DATA FF,4C,CC,FF,42,4F,4F,54
1040 DATA 01,A9,08,AA,A8,20,BA,FF
1048 DATA A9,02,A2,33,A0,C0,20,BD
1050 DATA FF,A9,00,20,D5,FF,90,03
1058 DATA 4C,E2,FC,20,AA,F5,86,2D
1060 DATA 86,2F,86,31,84,2E,84,30
1068 DATA 84,32,A9,00,20,59,A6,4C
1070 DATA AE,A7,20,20,XX
```

Paweł Sołtysiński

MAŁE, A CIESZY.

Często bardzo przydatne jest posiadanie czegoś takiego, jak znany ze Spectrum'a rozkaz BEEP, który generował dźwięki o żądanej wysokości i czasie trwania, i to wcale nie po to, aby wygrywać melodyjki. Czasami po żmudnych i czasochłonnych obliczeniach prowadzonych przez nasz program w BASIC'u dobrze byłoby poinformować o tym fakcie znużonego użytkownika, który właśnie zaczął zapadać w nerwową drzemkę.

Aby w prosty sposób zaradzić na brak takiego sygnału, postanowiłem napisać krótki programik, który wzorem GW Basic'a na komputery PC pozwalał by na generowanie krótkiego sygnału dźwiękowego przy drukowaniu znaku o kodzie ASCII równym 7. Po zainstalowaniu procedury (listing w języku BASIC może być z powodzeniem dołączony jako podprogram) sygnał dźwiękowy uzyskiwalny jest np. przez:

PRINT CHR$(7)

lub

PRINT „{CTRL+G}"

gdzie uzyskany znak jest kombinacją klawisza CTRL i litery G.

```
90  FOR T=272 TO 326:READ C:POKE T,C:NEXT:
    SYS 272
99  :
100 DATA 169,27,141,38,3,169,1,141,39,3,96,201,
    240,3,76,202,241,169,31,141
110 DATA 24,212,169,16,141,4,212,169,89,141,0,212,
    141,1,212,169,8,141,5,212,169
120 DATA 0,141,6,212,169,17,141,4,212,169,7,208,
    216
```

Paweł Sołtysiński

WSZYSTKICH ZAINTERESOWANYCH NABYCIEM ZALEGŁYCH NUMERÓW

„64 plus 4 & AMIGA”

INFORMUJEMY, ŻE POSIADAMY JESZCZE OGRANICZONĄ ILOŚĆ NUMERÓW

OD LISTOPADA 1990R. DO CZERWCA 1991R.

ZAMÓWIENIA PROSIMY KIEROWAĆ NA ADRES:
Przedsiębiorstwo ABUK sp. z o.o.,
87-200 Wąbrzeźno,
ul. 1 Maja 33.

(Pod tym adresem mieści się dział kolportażu - tam też prosimy przesyłać wszelką korespondencję dotycząca kolportażu czasopisma, dyskietek, taśm itd. Adres redakcji się nie zmienił - patrz stopka.)

ZAMÓWIONE NUMERY PRZEŚLEMY ZA ZALICZENIEM POCZTOWYM.

W związku z pojawiającymi się kłopotami w dystrybucji oferowanych przez nas dyskietek i taśm (wynikającymi z nieczytelnego bądź niekompletnego wypełnienia blankietów wpłat) przedstawiamy obok specjalny druk. Blankiet ten może służyć jako zamówienie i dowód wpłaty dla wszystkich oferowanych przez nas usług: sprzedaż dyskietek i taśm PDP, Voicetracker'a, zamówienie ogłoszeń itd.

REDAKCJA

UWAGA!

W związku ze wzrostem opłat pocztowych
(nasze pismo jest już grubsze i cięższe,
a więc musimy za jego wysyłkę płacić więcej)

z przykrością informujemy,
że od 1 sierpnia cena 1 egzemplarza „64 plus 4”
w prenumeracie wynosi 6000zł.

Przypominamy, że prenumeratę można zawrzeć na dowolny okres (nie krótszy niż dwa miesiące) do końca bieżącego roku. Na **CZYTELNIE** wypełnionych blankietach wpłat prosimy dopisać **„PRENUMERATA” oraz okres jej trwania.** Wpłaty prosimy przesyłać wykorzystując zamieszczony obok blankiet.

Przepraszamy.

Odcinek dla wpłacającego

Zł ....
słownie ....
wpłacający ....
....
....
(dokładny i CZYTELNY adres)

na rachunek:
Przedsiębiorstwa ABUK sp. z o.o.
87-100 Wąbrzeźno, ul. 1 Maja 33,
Bank PKO SA Bydgoszcz, konto:
5.09011-400522.7-136-11-111.0.

Opłata
zł..........

Odcinek dla Banku

Zł ....
słownie ....
wpłacający ....
....
....
(dokładny i CZYTELNY adres)

na rachunek:
Przedsiębiorstwa ABUK sp. z o.o.
87-100 Wąbrzeźno, ul. 1 Maja 33,
Bank PKO SA Bydgoszcz, konto:
5.09011-400522.7-136-11-111.0.

Opłata
zł..........

Odcinek dla posiadacza rachunku

Zł ....
słownie ....
wpłacający ....
....
....
(dokładny i CZYTELNY adres)

na rachunek:
Przedsiębiorstwa ABUK sp. z o.o.
87-100 Wąbrzeźno, ul. 1 Maja 33,
Bank PKO SA Bydgoszcz, konto:
5.09011-400522.7-136-11-111.0.

Opłata
zł..........

Odcinek dla Poczty

Zł ....
słownie ....
wpłacający ....
....
....
(dokładny i CZYTELNY adres)

na rachunek:
Przedsiębiorstwa ABUK sp. z o.o.
87-100 Wąbrzeźno, ul. 1 Maja 33,
Bank PKO SA Bydgoszcz, konto:
5.09011-400522.7-136-11-111.0.

Opłata
zł..........

TREŚĆ ZAMÓWIENIA:

Prosimy o CZYTELNE wypełnienie.

TREŚĆ ZAMÓWIENIA:

Prosimy o CZYTELNE wypełnienie.

TREŚĆ ZAMÓWIENIA:

PAMIĘTAJCIE!

Czytelne i dokładne wypełnienie wszystkich rubryk gwarantuje szybką i poprawną realizację Waszego zamówienia!

DZIĘKUJEMY!

W związku ze znacznym wzrostem opłat za przesyłki pocztowe, mając na uwadze dobro naszych czytelników sugerujemy zamawianie numerów zaległych naszego czasopisma na nieco innych zasadach.

Koszty przesyłki za tzw. zaliczeniu pocztowym przedstawia tabela (rubryki 2, 3, i 4). Wynika z niej, że koszty przesłania dwóch pierwszych numerów „64 plus 4" są większe niż ich wartość! Zamawiając numery wartości 6.000 zł zmuszeni jesteście dopłacić poczcie jeszcze 8.000 zł!

Zupełnie inaczej przedstawia się sytuacja w tabelach 5, 6 i 7. Wpłata kwoty z rubryki nr 7 na nasze konto (wraz z czytelną adnotacją, których numerów dotyczy, umieszczoną na wszystkich odcinkach blankietu) powoduje, że otrzymujecie przesyłkę bez kosztów pobrania!

Proponujemy abyście zaległe numery naszego pisma zamawiali w sposób następujący: wykorzystując tabelę - rubryki 1, 2, 5 i 7 - wyliczyli kwotę wpłaty, a następnie przesłali ją na nasze konto. Po otrzymaniu wpłaty natychmiast realizujemy przesyłkę!

Uwaga: dla dokonania wpłaty prosimy wykorzystać blankiet zamieszczony na tej stronie. Wszystkie dane prosimy pisać czytelnie i - zgodnie z rubrykami na blankiecie - **dziękujemy!**

Numery	Za pobraniem			Wpłata na konto		
	Wart. pisma	Koszt wysyłki	Razem	Wart. pisma	Porto	Razem
1	2	3	4	5	6	7
XI,XII	6.000,-	8.000,-	14.000,-	6.000,-	1.000,-	7.000,-
XI,XII,I	11.000,-	8.500,-	19.500,-	11.000,	1.500,-	12.500,-
XI,XII,I,II	16.000,-	8.500,-	24.500,-	16.000,-	1.500,-	17.500,-
XI,XII,I,II,III	21.000,-	8.500,-	29.500,-	21.000,-	1.500,-	22.500,-
XI,XII,I-IV	26.000,-	10.000,-	36.000,-	26.000,-	2.000,-	28.000,-
XI,XII,I-V	31.000,-	10.000,-	41.000,-	31.000,-	2.000,-	33.000,-
XI,XII,I-VI	36.000,-	10.000,-	46.000,-	36.000,-	2.000,-	38.000,-
XI,XII +I-VIII	46.000,-	14.000,-	60.000,-	46.000,-	3.000,-	49.000,-

Mini skaner

Profesjonalne skanery są bardzo drogie, dlatego wszystkim, którzy posiadają własną drukarkę proponujemy wykonanie prostego skanera we własnym zakresie. Tych, którzy wykonają to urządzenie oraz przygotują odpowiednie oprogramowanie prosimy o przesłanie, w postaci artykułu, opisu swôich rozwiązań i przykłady skanerowych obrazków. Najciekawsze prace postaramy się opublikować.

Proponowane przez nas rozwiązanie nie zapewnia profesjonalnej jakości, lecz do celów amatorskich jest wystarczające. Nasze urządzenie montowane jest na głowicy piszącej drukarki.

Sterując odpowiednio ruchem głowicy i wałkiem (należy dokładnie poznać kody sterujące posiadanej drukarki i wykorzystać je przy tworzeniu oprogramowania) krok po kroku źródło światła oświetla skanowany obrazek. Odbite światło, którego natężenie zależy od stopnia zaczernienia danego punktu obrazka, dociera do fototranzystora zmieniając jego rezystancję. Zmiany te są przetwarzane w komputerze na postać cyfrową (liczby od 0 do 255).

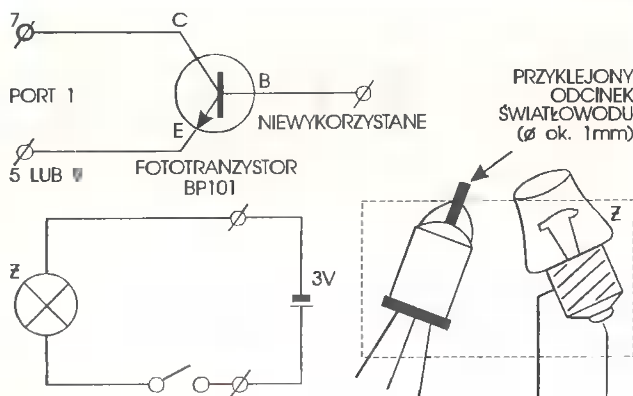
Tak uzyskaną informację należy już tylko odwzorować (przetworzyć) w punkt na ekranie monitora o odpowiednim stopniu szarości. Punkt po punkcie, linia po linii cały nasz obrazek zostaje zanalizowany i przekazany do pamięci komputera. Oczywiście przesuwem głowicy drukarki (wraz z naszym urządzeniem) oraz przetwarzaniem uzyskanych danych musi zarządzać odpowiednio napisany program.

Schemat naszego skanera przedstawia rys.1.

Żaróweczkę (najlepiej z soczewką w bańce - od miniaturowych latarek) i fototranzystor montujemy na wspólnym wsporniku, który będzie mocowany do głowicy drukarki. Konkretnych rozwiązań nie proponujemy ze względu na różnorodność kształtów głowic.

Generalnie należy pamiętać o dwóch zasadach:

> mocowanie skanera nie powinno w żaden sposób uszkodzić głowicy w drukarce - niedozwolone jest jej nawiercanie itp.



Rys. 1. Mini SKANER

> źródło światła i fototranzystor powinny być możliwie jak najbliżej skanowanego obrazka.

Korzystnym rozwiązaniem jest sklejenie żaróweczki i fototranzystora żywicą. Całość może być przytwierdzona do głowicy sprężynującą, blaszaną obejmą - pozwala to na łatwe montowanie i demontowanie skanera.

Należy zwrócić uwagę, aby cztery przewody (dwa doprowadzające zasilanie z baterii lub zasilacza do żaróweczki i dwa od fototranzystora) były odpowiednio cienkie i giętkie, by nie wprowadzały dodatkowych oporów i nie utrudniały ruchu głowicy w drukarce.

Fototranzystor podłączony jest (poprzez port 1 dla joystick'a) do wbudowanego w komputerze przetwornika analogowo-cyfrowego. Wpiszcie króciutki program:

```
10 A=54298
20 PRINT PEEK(A)
30 GOTO 20
```

Mając podłączony skaner uruchomcie program komendą RUN. Pojawiające się na ekranie cyfry będą się zmieniały w zależności od natężenia światła padającego na fototranzystor.

Dobierając kąt ustawienia żarówki i fototranzystora, odległość między nimi itd. możemy w znacznym stopniu wpływać na jakość analizy naszego obrazka.

Pole dla eksperymentów ogromne!

Opracowano na podstawie 64'er, nr 8/90

DISK MASTER V1.4

Program Disk Master służący do pracy ze zbiorami na dysku powstał bardzo dawno temu bo w roku 1987, ale jak do tej pory przewyższa on wszystkie inne programy do kopiowania zbiorów. Uruchamia się na własnym zadaniu oraz na własnym ekranie co czyni go bardzo wygodnym w użyciu zwłaszcza jeżeli dysponujemy dużą ilością pamięci.

Po uruchomieniu programu pojawia się ekran podzielony na dwie części, w każdej z nich będą znajdowały się nazwy zbiorów oraz wszelkie dane o tych zbiorach, które można odczytać z nagłówka. Wciskając lewy przycisk myszki na jednym z tych pól ustalamy je jako pole źródłowe (Source), a drugie automatycznie staje się polem przeznaczenia (Destination). Na górze każdego pola znajduje się nazwa katalogu, z którego są pobrane nazwy zbiorów znajdujące się w polu. Na górnym pasku ekranu (tzw. Menu Bar) znajdują się informacje o wolnej pamięci, ilości wolnych bajtów w aktualnych katalogach oraz nazwy dysków z kórych pochodzą wybrane katalogi. Oprócz tego znajdują się tam trzy gadżety (pod KickStartem 2.0 tylko dwa), a mianowicie gadżet do zamknięcia okna - zakończenie pracy oraz gadżety (bądź gadżet) głębokości.

Na środku ekranu znajduje się pasek gadżetów służących do pracy na zbiorach. Pierwsze sześć (licząc od góry) służy do wybrania odpowiedniego katalogu i są to pola definiowalne (patrz Dodatek A) dodatkowe sześć pól otrzymujemy wciskając prawy przycisk myszki na jednym z tych sześciu pól. Otrzymamy pola dodatkowe, które możemy także zdefiniować.

Do pierwszego zestawu wracamy wciskając drugi raz przycisk myszki. Chcąc uzyskać któryś z podanych katalogów wskazujemy go kursorem myszki. W przypadku nieznalezienia go na górnym pasku pojawi się napis „Bad directory". Gdy program wczyta dany katalog umieści go w oknie wybranym poprzednio jako źródłowe.

Kolejnymi gadżetami są:

> **Parent** - wczyta poprzedni katalog (tzw. rodzic), znaczy to, że gdy mamy na przykład wczytany katalog „DF0:C" to po opcji „parent" bdziemy mieli katalog „DF0:".

> **All** - wszystkie zbiory znajdujące się w oknie wybranym jako źródłowe zostaną wybrane. Wyboru poszczególnych pozycji katalogu dokonujemy poprzez wciskanie przycisku myszki „na nich", a wejścia do podkatalogów dokonujemy poprzez podwójne wciśnięcie lewego przycisku myszki. Możemy także uzyskać podkatalog w drugim polu przeznaczonym na zbiory. Najpierw wybieramy odpowiedni katalog poprzez wciśnięcie przycisku myszki gdy kursor znajduje się na jego nazwie, a następnie przesuwamy kursor ponad drugie pole i wciskamy lewy przycisk myszki.

> **Clear** - wszystkie pozycje zostaną zdeselekcjonowane.

> **Copy** - kopiuje wszystkie wybrane zbiory i katalogi z pola źródłowego do katalogu wybranego w polu przeznaczenia.

> **Rename** - dzięki tej opcji możemy zmieniać nazwy zbiorów i katalogów wybranych w polu źródłowym.

> **Move** - w działaniu podobna do Copy jednak po wykonaniu kopii wszystkie zbiory zostaną usunięte z katalogu źródłowego.

> **Delete** - usunięcie wybranych zbiorów.

> **Comment** - zmiana bądź dopisanie komantarzy do wybranych zbiorów.

> **Protect** - ustawi odpowiednio bity protekcji wybranych zbiorów (Dodatek B).

> **Search** - szuka w danym katalogu zbiorów o podanej nazwie. Możemy używać skrótów „*", np. *.pp - odnajdzie wszysktkie zbiory z końcówką „pp".

> **Read** - pozwala na odczyt zbiorów w postaci znaków ASCII. Gdy wciśniemy na tym gadżecie prawy przycisk myszy zmienia on się na gadżet „HexRead" służący do oglądania zbiorów w postaci AHex (czyli kody szesnastkowe i znaki ASCII).

> **Print** - pozwala na wydruk wybranych zbiorów.

> **ShowPic** - pozwala na oglądanic rysunków oraz odtwarzanie sampli zapisanych w formacie IFF. Po wciśnięciu prawego przycisku zmienia się na „PlaySnd" i pozwala na traktowanie zbiorów jako danych dla sampli.

> gadżet komend definiowanych przez użytkownika. Możemy zdefiniować aż 10 takich komend, a wyboru odpowiedniej dokonujemy prawym przyciskiem myszki (Dodatek C).

> **MakeDir** - służy do wykreowania podkatalogu w katalogu wskazywanym przez pole źródłowe.

Wciskając prawy przycisk myszki mamy dostęp do menu.

Menu Project:

- **Disk Copy** - kopiowanie całych dysków,
- **Format** - formatowanie dysków,
- **Print Dir** - wydruk katalogu,
- **Workbench** - otwiera i zamyka workbench,
- **About** - informacja o programie.

Menu Configure:

- **Set Colors** - pozwala na ustawienie kolorów dogodnych dla użytkownika.
- **Resolution** - pozwala na wybór rozdzielczości.
  Lo-Res oznacza ekran w rozdzielczości 640*256 punktów.
  Hi-Res - ekran 640*512 punktów (włączony interlace).
  Half height - ekran 640*256 (plus interlace) o wysokości połowy normalnego ekranu.
- **Small Font** - zastosowanie mniejszej czcionki.

- **Auto Dir** - wczytywanie katalogu bezpośrednio po włożeniu dyskietki w urządzenie.
- **Info Copy** - kopiowanie ikonek wraz ze zbiorami
- **Confirm** - program będzie prosić o potwierdzanie niebezpiecznych operacji.
- **Set Protect** - patrz Dodatek B.

Menu odpowiadające za komendy użytkownika (patrz Dodatek C);

- **Set Device** - patrz Dodatek A.
- **Save Config** - nagranie ustawionej konfiguracji programu Disk Master na dysk.

Menu Archive

Służy ono do współpracy z programami archiwizującymi takimi jak: Arc, Zoo, LHArc. Są tam opcje „add", „extract", „list" służące do archiwizacji oraz dearchiwizacji zbiorów.

Aby korzystać z programów archiwizujących oraz komend użytkownika musimy w katalogu C posiadać programy archiwizujące oraz komendy „Run", „CD", „NewCLI", „EndCLI".

Dodatek A

Aby ustawić własne katalogi w gadżetach musimy wybrać opcję Set Device z menu Configure, a następnie wybrać odpowiedni gadżet oraz wpisać nazwę katalogu, która ma być wywoływana za pośrednictwem tego gadżetu, np. „DF0:", „DH0:", „VD0:", „RAD:", „DF2:LIBS", itp. Musimy jednak pamiętać aby nie przekroczyła ona ośmiu znaków.

Dodatek B

Dzięki opcji z menu Configure: Set Protect możemy ustawić stałą maskę bitów protekcji ustawianych w zbiorach za pomocą opcji Protect. Te bity to HSPARWED, przy czym tylko bity WED są uznawane przez system Amigi i służą do protekcji zbiorów przed zapisem „W", edycji „E", skasowaniem „D".

Dodatek C

Komendy użytkownika mogą być ustawione poprzez wybranie ich (prawy przycisk na gadżecie komend użytkownika), a następnie wybraniu z menu opcji ustawienia komendy (będzie ta opcja miała taką nazwę jak aktualna komenda użytkownika - patrzymy na gadżet z tą komendą - w Disk Masterze bez pliku konfigurującego są to: Cmd 1, Cmd 2, itd. Teraz możemy wprowadzić różne rzeczy. W mojej kopii DiskMastera wprowadziłem sobie takie komendy, jak: Run, PPMore, PPShow, PPType, PPAnim, itp. Aby wprowadzić komendę tego typu musimy wpisać jakąś nazwę komendy np. „PPMore %s" znak „%s" (znany programującym w języku C) służy do tego, aby jako parametr dla komendy (PPMore) zostały podane nazwy wybranych zbiorów.

Podobno pojawiła się wersja 2.0 Disk Master'a ale jej jeszcze nie posiadam. Już dosyć dawno, krążyła wersja 3.0, ale był to zupełnie inny program napisany przez zupełnie innego programistę i oczywiście posiadał wiele niedociągnięć.

Marcin "Duddie" Dudar

SCULPT ANIMATE 4D cz.5

Nasz cykl artykułów o Sculpt'cie rozpoczęliśmy trochę od końca, ale teraz właśnie dochodzimy do początku, czyli menu Project.

MENU PROJECT składa się kolejno z następujcych pod-menu:

Load
Save
Show
Unload
Batch
About
Quit

Menu: Load

Scene	- wczytanie z dysku odpowiednio przygotowanej sceny czyli wszystkich obiektów, źródeł światła, obserwatora, itp.
Image	- wczytanie gotowego obrazka
Object	- wczytanie przygotowanego obiektu i umieszczenie go na środku okna edycji.
Named object	- wczytanie nazwanego obiektu
Animation	- wczytanie z dysku przygotowanej animacji
Font	- wczytanie czcionki dla programu
Sculpt	- podajemy tylko nazwę katalogu, w którym dana czcionka się znajduje
Code	- wczytanie całego programu Sculpt do pamięci
Workbench	- otworzenie Workbencha

Menu: Save

Scene	- nagranie aktualnej sceny na dysk
Image	- nagranie wygenerowanego rysunku
Object	- nagranie obiektu ktrógo wierzchołki są aktualnie aktywne
Named object	- nagranie nazwanego obiektu - patrz menu Edit (Named object)

Menu: Show

Image	- pokazanie wygenerowanego rysunku
Animation	- pokazanie wygenerowanej animacji przez program Sculpt 4D
Preview	- podgląd animacji

Menu: Unload

Image	- usunięcie z pamięci gotowego obrazka
Animation	- usunięcie gotowej animacji
Preview	- usunięcie podglądu
Font	- usunięcie czcionek
Code	- usunięcie części programu Sculpt aby otrzymać więcej pamięci do pracy
Workbench	- zamknięcie Workbencha

c.d.n.

Marcin „Duddie" Dudar

KURS JĘZYKA C (cz.1)

Amiga DOS powstał w jezyku C i ten fakt od razu tłumaczy, dlaczego przedstawiamy Wam akurat ten język programowania. Jego poznanie pozwoli automatycznie zapoznać sie z organizacją systemu operacyjnego. A poza tym język C jest przez specjalistów uznawany zgodnie za język lat dziewięćdziesiątych - tak, jak Pascal w latach osiemdziesiątych.

Najpierw kilka uwag ogólnych. Wszystkie informacje i przykłady będą działały zarówno na kompilatorach Aztec C i SAS-Lattice C. Ewentualne różnice będą wyjaśniane w tekście opisu.

O ile mamy dostateczną ilość pamięciprzy pracy z kompilatorami polecam korzystanie z ram-dysku. Przyspiesza to czas kompilacji programów, a w przypadku użytkowników twardych dysków skuteczenie chroni ich przed koniecznością kupowania po paru miesiącach nowych napędów.

Kompilatory na Amigę rażą niestety ubóstwem rodem z epoki ODRY (kto pracowal kiedyś na kompilatorach serii Turbo firmy Borland na IBM'ie ten wie, o co chodzi). Komfort pracy jest tu doprawdy minimalny, a błędy typu 24, 43, 74 i tym podobne „czytelne" komunikaty to chleb powszedni. Wypada mieć tylko nadzieję, że w przyszłości kompilatory serii Turbo pojawią się na Amigę.

Standardowo pierwszym programem, jaki pisze się w nowym języku, jest wydruk - chcemy np. wydrukować tekst „AMIGA? AMIGA!!!". W tym celu powinniśmy wprowadzić następujacy tekst programu:

```
main()
{
 printf ("AMIGA? AMIGA!!!");
}
```

Zanim opiszę działanie tego programu należy powiedzieć kilka słów o konwencji zapisu tekstu programu. Otóż programy powinny być pisane czytelnie, jasno i „chlujnie" (w przeciwieństwie do programów niechlujnych).

Otóż żelazną zasadą jest, że KAŻDY nawias klamrowy {} (oznaczają one jakiś blok programu - np. pętle) powinien być pisany w nowej linii. Dobrze jest kolejny taki nawias umieścić np. po dwóch spacjach. Wtedy bardzo łatwo można stwierdzić, ile nawiasów otwieraliśmy i ile zamykaliśmy.

Np.

```
{          /* główny tekst */
 {         /* tekst 1 */
  {        /* tekst 2 */
  }        /* tekst 2 */
 {         /* tekst 1 */

 {         /* tekst 3 */
  {        /* tekst 2 */
  }        /* tekst 2 */
 }         /* tekst 3 */
}          /* glowny tekst */
```

Każdy rozkaz programu powinien znajdować się w nowej linii. Również nowy blok programu (tj. część zajmująca się czymś innym) dobrze jest pisać od nowej linii. Pisząc pętle piszemy od razu dwa nawiasy: {} a dopiero potem treść pętli - dzięki temu nie zapomnimy zamknąć pętli. Te zasady są żelazne - dzięki nim program jest czytelny i łatwo go modyfikować.

Jasno napisany program zawiera komentarze w kluczowych miejscach, próżno szukać w nim niewykorzystanych zmiennych, fragmentów programu które nic nie robią i różnych innych pozostałości po bolesnym porodzie tekstu programu.

Naprawdę warto się przejąć tymi zasadami BHP i stosować je od samego początku!

Mam nadzieję, że przekona was ten sam przykład z nawiasami co powyżej, napisany trochę inaczej:

```
{
{
}
{
}
{
}
{
}
}
```

Wróćmy do programu drukującego tekst. Każdy program w języku C składa się z funkcji, a każda z funkcji może mieć praktycznie dowolną nazwę (z drobnymi wyjątkami). Jednym z takich wyjątków jest funkcja **main**. Od niej właśnie program rozpoczyna działanie. Treść funkcji **main** zawarta jest pomiędzy nawiasami

klamrowymi i obejmuje w naszym wypadku jedną funkcję - **printf.**

> **UWAGA!**
> W języku C oprócz słów kluczowych języka (typu np. **main**) występują standardowe funkcje. Jedną z takich standardowych funkcji wejścia/wyjścia jest właśnie funkcja **printf**.

Każda funkcja w C bezpśrednio po swojej nazwie ma umieszczone w nawiasach zwykłych () swoje argumenty. W przypadku funkcji **main** nie mieliśmy żadnych argumentów, lecz nawiasy muszą pozostać! Argumentem funkcji **printf** był natomiast tekst „AMIGA? AMIGA!!!". Argument ten umieszczony został w dwóch nawiasach zwykłych. Po każdej funkcji, wyrażeniu itp. musi wystapić średnik oznaczający koniec wyrażenia - (stąd też średnik jako ostatni znak po nawiasie zamykającym argumenty funkcji **printf**). Ponieważ bez **printf** nic nie zrobimy, zajmijmy się tą funkcją bliżej.

Printf jest funkcją, która swój argument - np. tekst - drukuje od aktualnej pozycji kursora na ekranie. Stąd też by uzyskać wydruk tekstu od nowej linii należy na początku tekstu (pomiędzy cudzysłowami oznaczającymi „stałą tekstową") wpisać sekwencję:

\n

Sekwencja ta oznacza nową linię. Spróbujmy wypisać różne teksty, każdy zaczynający się od nowej linii. Teraz nasz program ma postać:

```
main()
{
  printf ("\nAMIGA? AMIGA!!!");
  printf ("\nJezyk C nie jest taki straszny");
  printf ("\nNA JAKI WYGLADA!!!");
}
```

Zauważmy, że po każdej funkcji następuje średnik i że każda stała tekstowa rozpoczyna się od znaku nowej linii. Sekwencja **\n** reprezentuje jeden znak - końca linii - i jest nazywana sekwencją specjalną. Inne sekwencje specjalne to:

\t - znak tabulacji
\b - cofnięcie o jeden znak (backspace)
\" - znak cudzysłowia
\\ - po prostu znak \

Sprawdzenie jak one działają w programie pozostawiam inwencji Czytelników.

Każdy język programowania pozwala na operowaniu zmiennymi. W języku C mamy zmienne różnych typów: **char** (jeden znak), **short** (liczba całkowita krótka), **long** (liczba całkowita długa), **int** (całkowite), **float** (zmiennoprzecinkowe), **double** (zmiennoprzecinkowe podwójnej precyzji).

Oto krótka tabelka zawierająca informacje o tym, jakie typy zmiennych mają rozmiary:

Typ	Rozmiar	Wartość minimalna	Wartość maksymalna
char	8	-128	127
unsigned char	8	0	255
short	16	-32768	32767
unsigned short	16	0	65535
int	32	-2147483648	21474836647
unsigned int	32	0	4294987295
long	32	-2147483648	2147483647
unsigned long	32	0	4294987295
float	32	10E-37	10E+38
double	64	10E-307	10E+308

Wartości minimalne i maksymalne zmiennych typu float i double zostały podane orientacyjnie (zapisy te oznaczają 10 do potęgi -37, 10 do potęgi 38 itp.).

Nie ze wszystkich typów zmiennych będziemy korzystać od razu.

```
main()
{
  short jeden;
  unsigned short dwa;

  jeden = -2354;
  dwa = 53257;

  printf ("\nOto jeden = %d,\na to dwa = %d", jeden, dwa);
}
```

W tym krótkim programie zwróćmy uwagę, że zaraz na początku następują deklaracje zmiennych, jakie będą użyte w programie - jeden i dwa. Następnie zmiennym tym nadajemy określone wartości. Istotne jest tu przestrzeganie zakresów dozwolonych dla każdej ze zmiennych - eksperymentatorom polecam zamianę wartości zmiennych: (jeden = 53257, a dwa = -2354). Funkcja **printf** dotychczas miała tylko jeden argument - stałą tekstową. Teraz ma trzy!

Przeczytajmy te argumenty. "\nOto jeden = %d,\na to dwa = %d" oznacza: od nowej linii (**\n**) wypisz tekst "**Oto jeden =**", nastepnie wydrukuj wartość pierwszej zmiennej całkowitej występującej jako argument **printf** (to mówi znak: **%d**), następnie wydrukuj przecinek, od nowej linii tekst "**a to dwa =**" po czym wartość zmiennej występującej jako druga w argumencie **printf** (znak **%d**). Kolejnym argumentem **printf** jest nazwa zmiennej, która jako pierwsza ma być wydrukowana (w miejsce

pierwszego znaczka **%d**), a więc **jeden**, po czym mamy wypisaną zmienną o nazwie **dwa**, która będzie umieszczona w miejscu drugiego znaczka **%d**.

Oto niektóre inne znaki przekształceń:

%d - zmienna będzie przekształcona do postaci dziesiętnej (long, short itp.),

%o - zmienna będzie przekształcona do postaci ósemkowej bez znaku (bez początkowego zera),

%x - zmienna będzie przekształcona do postaci szesnastkowej bez znaku (bez początkowego 0x),

%u - zmienna będzie przekształcona do postaci dziesiętnej bez znaku,

%c - zmienna będzie traktowana jako jeden znak (char),

%s - zmienna będzie traktowana jako tekst ,

%e - zmienna jest traktowana jako float lub double i przekształcona do postaci dziesiętnej [-]m.nnnnnnE[+-]xx,

%f - zmienna jest traktowana jako float lub double i przekształcona do postaci dziesiętnej [-]mmm.nnnnnn

%g - użycie %e lub %f w zależnosci od tego, która daje krótszy tekst.

Nie przejmujmy sie zbytnio tym, że część tych informacji jest niezrozumiała. Wątpliwości wyjaśnią się stopniowo w czasie lektury.

Wróćmy do przykładu.

Zauważmy, że deklaracje zmiennych składają sie ze słowa określajacego typ zmiennej (**short, float** itp.) i nazwy zmiennej.

UWAGA!
Język C rozróżnia duże i małe litery, dlatego też zmienna **Jeden** i zmienna **jeden** to dwie zupełnie różne zmienne. Zmienna może mieć dowolną nazwę inną jednak od nazw słów kluczowych, nazw standardowych funkcji itp. Dobrze jest zmiennej nadawać znaczące nazwy, np. dla zmiennej używanej w pętlach - loop.

UWAGA!
Jeszcze raz podkreślam konieczność zwracania dużej uwagi na zakresy zmiennych. Osobiście zdażyła mi się sytuacja, gdzie w dużym programie (ok. 5000 linii) zmienna loop miała rozmiar 8 bitów - maksymalną wartość 255. W jednym przypadku w pętli zawarte było wyrażenie „wykonuj dopóki loop mniejsze od 256". Jak się domyślacie, dość długo czekałem na efekt działania tej pętli...

Po każdej deklaracji następuje, jak zwykle, średnik. Dobrze jest oddzielać deklaracje od reszty programu jedną lub nawet dwoma wolnymi liniami. Następnie w programie nadajemy zmiennym ich wartości - ta inicjalizacja jest konieczna, bowiem zmienne, które są tylko zadeklarowane, mogą mieć nieustalone wartości.

O funkcji **printf** i sposobach wydruku zmiennych pomówimy jeszcze w następnych odcinkach.

Narazie jeszcze jeden przykład:

```
main()
{
 int Pierwsza;
 short Druga;
 float Trzecia;
 double Czwarta;
 char Piata;

 Pierwsza = 10000000;
 Druga = 1000;
 Trzecia = 3.14159;
 Czwarta = 2.71828;
 Piata = 'a';

 printf ("\nOto int = %d, oto short = %d", Pierwsza,
Druga);
 printf ("\nOto float = %f, oto double %e", Trzecia,
Czwarta);
 printf ("\nOto char = %c", Piata);
}
```

Ten program używa prawie wszystkich typów zmiennych i drukuje je na ekranie. Warto zwrócić uwagę na sposób inicjalizowania zmiennej typu char - wartość tej zmiennej (jeden znak!) ujmuje się w znaki ' '.

Aby naprawdę docenić uroki języka C należy na początku, niestety, przekopać się przez te wszystkie nudnawe, ale potrzebne informacje. Nawet jeżeli coś jest na razie niejasne i niezrozumiałe, powinno się wyjaśnić w praktyce i oczywiście podczas lektury kolejnych części cyklu.

Jarosław Chrostowski

KĄCIK POCZĄTKUJĄCEGO KODERA (cz.7)

Dzisiaj poznamy kolejne rozkazy mikroprocesora 68000, a potem program mający ścisły związek z dwoma ostatnimi artykułami z cyklu „Jak zrobić własne demo".

- **DIVS adres efektywny, Dx „Signed Divide" - dzielenie ze znakiem.**

Instrukcja ta dzieli 32-bitową liczbę zawartą w dowolnym rejestrze danych przez 16-bitową określoną adresem efektywnym dając 32-bitowy wynik, przy czym górne 16 bitów rejestru danych jest resztą z dzielenia, a dolne ilorazem. Instrukcja ta pracuje w arytmetyce ze znakiem tzn. najstarszy bit interpretowany jest jako znak.

Przy wykonywaniu tej instrukcji popełnione mogą być dwa błędy:

1. Dzielimy liczbę „dużą" przez „małą" i iloraz nie może (z uwagi na to, iż jest zbyt duży) zostać poprawnie zapisany na 16-bitach. W tej sytuacji zostanie ustawiony znacznik V, gdy spenione są warunki nadmiaru.
2. Dzielimy dowolną liczbę przez 0. W takiej sytuacji procesor generuje stan wyjątkowy.
   Należy pamiętać iż dzielenie jest jedną z najdłużej wykonywujących się instrukcji - dlatego też radzę używać jej jedynie w ostateczności. O ile to możliwe starajmy się ją zastępować instrukcjami przesunięć i obrotów, oraz mechanizmami tablicowania.

Znaczniki:
X - nie zmieniany
N - ustawiany zgodnie z najstarszym bitem wyniku.
Z - ustawiany, gdy iloraz jest równy 0
V - ustawiany, gdy iloraz nie może być wyrażony poprawnie
C - zawsze zerowany

Przykład:

divs #-5,d1 - rejestr danych d1 zostanie podzielony przez -5. Wynik zostanie zapisany na młodszym słowie rejestru d1, reszta z dzielenia na starszym.

- **DIVU adres efektywny,Dx „Unsigned Divide" - dzielenie bez znaku.**

Diałanie tej instrukcji jest analogiczne do DIVS, z tą różnicą, iż DIVU działa w arytmetyce bez znaku.

Znaczniki: analogicznie jak DIVS

Przykład:

divu d0,d1 - rejestr d1 zostanie podzielony przez młodsze słowo rejestru d0.

- **NEG adres efektywny "Negate" - negacja.**

Instrukcja ta neguje liczbę określoną adresem efektywnym. W wyniku jej działania otrzymamy liczbę o takiej samej wartości bezwzględnej, ale o przeciwnym znaku. Negację można przeprowadzać na operandach wielkości bajtu, słowa lub długiego słowa.

Znaczniki:
X - zerowany, gdy wynik jest równy zero, w przeciwnym wypadku ustawiany
N - ustawiany, gdy po operacji negacji otrzymamy liczbę ujemną
Z - ustawiany, gdy wynik jest równy 0
V - ustawiany w przypadku nadmiaru
C - tak samo jak X

Przykład:

move.l #23,d0

neg.ld0 - w wyniku wykonania tej sekwencji otrzymamy w d0 wynik równy -23

- **NEGX adres efektywny "Negate With Extend" - negacja z rozszerzeniem.**

Wykonuje negację liczby określonej adresem efektywnym wykorzystując bit rozszerzenia X. Rozmiar operacji: bajt, słowo lub długie słowo.

Znaczniki:
X - ustawiany w przypadku wygenerowania pożyczki
N - ustawiany, gdy liczba jest ujemna
Z - ustawiany, gdy wynik jest równy zero
V - ustawiany, gdy wystąpi nadmiar
C - ustawiany w przypadku wygenerowania pożyczki.

Przykład:

negx.b d0

- **ILLEGAL „Illegal" instrukcja niedozwolona.**

Instrukcja ta nie powoduje żadnego działania. Mikroprocesor po jej napotkaniu wygeneruje stan wyjątkowy nielegalnej instrukcji. Używana czasem przez koderów przy uruchamianiu programów i ich testowaniu.

Znaczniki: nie są modyfikowane.

Przykład:

illegal

- **EOR Dx, adres efektywny „Exclusive OR" - logiczne „albo".**

Instrukcja EOR wykonuje logiczne „albo" rejestru danych z liczbą określoną adresem efektywnym. Operacja może przybierać rozmiar bajtu, słowa lub długiego słowa. Logiczne „albo" przyjmuje wartość 0 jeżeli oba zdania posiadają tak samą ocenę logiczną.

Przedstawia to poniższa tabela:

0	EOR	0	=	0
0	EOR	1	=	1
1	EOR	0	=	1
1	EOR	1	=	0

Instrukcja EOR często używana jest do kodowania przgramów, oraz wszelkiego rodzaju zabezpieczeń.

Znaczniki:
X - Nie zmieniany
N - Ustawiany, gdy wynik jest ujemny
Z - Ustawiany, gdy wynikiem jest 0
V - Zerowany
C - Zerowany.

Przykład:

```
move.b  #%10011101,d0
move.b  #%01010101,d1
eor.b   d1,d0          - w wyniku wykonania tej
                       ;sekwencji otrzymamy liczbę
                       ;%11001000 umieszczon w d0
```

- **EORI #liczba, adres efektywny „Exclusive OR Immediate" - natychmiastowe „albo" logiczne.**

Analogiczne działanie do EOR, z tą różnicą, że logiczne „albo" wykonywane jest z daną natychmiastową.

Znaczniki: jak EOR

Przykad:

```
eor.w    #$acb1,d0
```

Przedstawiony poniżej program ma ścisły związek z artykułami publikowanym w poprzednim i dzisiejszym numerze pt. „Sprites". Dlatego zanim przystąpicie do jego analizy propnuję ponownie przeczytać artykuł z ostatniego numeru i zapoznać się z treścią artkułu zamieszczonego w tym numerze.

Program zamieszczony poniżej: inicjuje i umieszcza na ekranie sprite'a demonstrując jego podwójne zastosowanie. Jedna część sprite'a zostaje następnie w prosty sposób animowana. Nie chodziło mi tu o pokazanie sprite'ów płynnie poruszających się po wyszukanych krzywych lecz jedynie o zaprezentowanie mechanizmów animacji, a czytelnikowi pozostawiam rozbudowanie tej procedury o ruch pionowy, lub po sinusie co daje zaskakujące efekty.

```
start:  move.l  #copper,$dff080     ;Inicjacja Copper'a
        move.l  #sprite0,d0         ;Inicjacja Sprite'a 0
        move.w  d0,meml0
        swap    d0
        move.w  d0,memh0
loop:   cmp.b   #$ff,$dff006
        bne     loop
        move.w  #100,d0             ;Potrzebne pętli
                                    ;opóźniającej
        addq.b  #1,PosX             ;Zwiększamy pozycję
                                    ;poziomą ruch w prawo
        cmp.b   #200,PosX           ;Gdy dojdzie do 200
                                    ;ustawiamy na 70
        bne     cont
        move.b  #70,PosX
cont:   nop                         ;Pętla opóźniająca -
                                    ;opóźnienie w d0
        dbf     d0,cont
        btst    #6,$bfe001          ;Test LMB
        bne     loop
        moveq   #0,d0
        rts
copper: dc.l $00968020              ;Włącześnie
                                    ;DMA
        dc.l $01080000,$010a0000    ;Modulacja
        dc.l $01000000,$01020000,$01040000 ;Rejestry
                                    ;kontroli
        dc.l $008e2c50,$00902cc1,$00920038,$009400d0
                                    ;Definicja
                                    ;obrazu
        dc.l $00e00007,$00e20000    ;Obszar
                                    ;pamięci
                                    ;obrazu
        dc.l $01800000,$01820fff    ;i kolory dla
                                    ;niego
        dc.l $01a20f00,$01a400f0,$01a6000f ;Kolory
                                    ;sprite'a 0
        dc.w $0120                  ;Tu zostanie
                                    ;wpisany
                                    ;adres
                                    ;struktury
memh0:  dc.w $000                   ;danych dla
                                    ;sprite'a 0
        dc.w $0122
meml0:  dc.w $0000

        dc.l $3501fffe              ;Włączenie bit
                                    ;plane'u
                                    ;konieczne,
        dc.l $01001200              ;gdyż sprite'y
                                    ;nie mogą
                                    ;występować
        dc.l $f801fffe              ;przy
                                    ;wyłączonych.
        dc.l $01000000
        dc.l $fffffffe              ;Koniec
                                    ;CopperList'u
Sprite0:        :     ****Struktura Danych Sprite'a 0****
        dc.b    $60   ;VSTART    Ten i następny bajt stanowi
PosX:   dc.b    $89   ;HSTART    pierwsze sowo kontroli
        dc.b    $68   ;VSTOP     Tu zaczyna się drugie
                                 słowo kontroli
        dc.b    %00000000        ;CTRL
dc.w    %1111111111111111,%0000000000000000 ;Kształt
                                            ;i kolory
dc.w    %1100000000000011,%0011111111111100
dc.w    %1100111111110011,%0011111111111100
dc.w    %1100110000110011,%0011110000111100
dc.w    %1100110000110011,%0011110000111100
dc.w    %1100111111110011,%0011111111111100
dc.w    %1100000000000011,%0011111111111100
dc.w    %1111111111111111,%0000000000000000
dc.b    $80       ;VSTART        ;Ponowne użycie sprite'a 0
dc.b    $70       ;HSTART
dc.b    $84       ;VSTOP
dc.b    %00000000 ;CTRL
dc.w    %011100001110,%0000011101110
dc.w    %011100001110,%0000011101110
dc.w    %011100001110,%0000011101110
dc.w    %011100001110,%0000011101110
dc.w    $0000,$0000              ;Koniec struktury danych
                                 ;dla sprite'a
```

Krzysztof "K.K." Kobus

ARP LIBRARY (CZ.1)

Biblioteka ARP (Amiga DOS Replacement Project) powstała, aby zastąpić przestarzałą bibliotekę „dos.library". Jej twórcy chcieli stworzyć taką bibliotekę, której użycie zamiast dos. library nie zmuszałoby użytkownika do gruntownej zmiany programu. Można powiedzieć, że udało im się zrealizować ten cel wyśmienicie. Oprócz standardowych funkcji dos.library posiada ona także dużo nowych, bardzo potrzebnych funkcji, np. Resident, Assign i wiele, wiele innych.

Biblioteka ta jest kompatybilna zarówno ze starym standardem w jakim została napisana dos.library, a więc BCPL oraz z nowym jaki wnosi język C. Ze względu na dużą objtość tej biblioteki, nasz wykład zostanie podzielony na kilka części, a przykłady zostaną podane później.

Procedury będą podawane w kolejności użytkowania i dzisiaj słów kilka o programach rezydentnych.

LoadPrg - wczytanie programu do pamięci
Segment = LoadPrg("Nazwa")
D0 D1

Funkcja:

Wczytanie programu do pamięci ale w inny sposób niż to robi procedura LoadSeg z biblioteki dos.library. Procedura ta w pierwszej kolejności przeszukuje listę programów rezydentnych, a następnie katalog aktualny oraz katalog „C:" i katalog systemowy „SYS:".

Wejście:

Nazwa - adres nazwy programu, który chcemy załadować zakończonej zerem.

Wyjście:

Segment - adres segmentu w formacie BCPL (tzw. BPTR czyli BCPL Pointer) czyli rzeczywisty adres dzielony przez cztery. W przypadku nie znalezienia programu lub innego błędu w D0 otrzymamy wartość 0. Po wywołaniu procedury IoErr powinniśmy otrzymać właściwy numer błędu.

UnLoadPrg - usunięcie z pamici programu wczytanego przez LoadPrg
UnLoadPrg(Segment)
D1

Funkcja:

Procedura zwalnia miejsce w pamięci zajmowane przez program wskazany przez wskaźnik, Segment jednak najpierw przeszukuje listę programów rezydentnych aby nie usunąć programu aktualnie używanego.

Wejście:

Segment - wskaźnik BPTR do struktury Segment (otrzymany z LoadPrg)

Wyjście:

Nic.

AddResidentPrg - dodanie programu do listy programów rezydentnych
Node = AddResidentPrg(Segment, Nazwa)
D0 D1 A0

Funkcja:

Procedura ta pozwala na dodanie danego programu do listy programów rezydentnych czyli takich, które mogą być uruchamiane kilka razy z jednej kopii znajdującej się w pamięci bez potrzeby wczytywania ich na nowo.

Wejście:

Segment - BPTR Segment, który otrzymujemy z procedury LoadSeg lub LoadPrg.

Nazwa - adres nazwy zakończonej zerem, która będzie przypisana do tego prgramu (nie musi być to oryginalna nazwa tego programu!)

Wyjście:

Node - wskaźnik dla nagłówka rezydentnego programu. Jeżeli procedura zwróci zero oznacza to, że nie ma możliwości na dodanie programu do listy programów rezydentnych (np. zdublowana nazwa). Informacje dokadniejsze pod procedura IoErr.

RemResidentPrg - usunięcie programu z listy programów rezydentnych
Użycie = RemResidentPrg(„Nazwa")
D0 A0

Funkcja:

Procedura usuwa program z listy programów rezydentnych i zwalnia pamięć zajmowaną przez jego nagłówek oraz przez sam program.

Wejście:

Nazwa - wskaźnik zakończonej zerem nazwy programu, który chcemy usunąć.

Wyjście:

Użycie - jeżeli wartość jest różna od zera to oznacza ile procesów wykorzystuje nasz program i z tego powodu nie może on być usunięty. Gdy jest to wartość zero oznacza, że operacja się powiodła.

ObtainResidentPrg - otrzymanie dostpu do programu rezydentnego
Node = ObtainResidentPrg(„Nazwa")
D0 A0

Funkcja:

Procedura znajduje program rezydentny o zadanej nazwie i zwiększa licznik uruchomień tego programu aby nie mógł być on usunięty.

Wejście:

Nazwa - wskaźnik zakończonej zerem nazwy programu.

Wyjście:

Node - wskaźnik struktury rezydentnego programu. Nie jest to struktura Segment, ale można strukturę Segment znaleźć wewnątrz struktury Node. W przypadku otrzymania zera wystpił błąd i możemy otrzymać więcej informacji po wywołaniu procedury IoErr.

ReleaseResidentPrg - zwolnienie przez użytkownika dostępu do programu rezydentnego
Node = ReleaseResidentPrg(Segment)
D0 D1

Funkcja:

Po powrocie z programu rezydentnego użytkownik powinien zwolnić dostęp do niego. Procedura zmniejszy licznik uruchomie programu.

Wejście:

Segment - wskaźnik załadowango segmentu.

Wyjście:

Node - wskaźnik struktury node nie powinien być używany za wyjątkeim testu True/False (Prawda/Fałsz) gdyż jest to już nieaktywny program.

BaseName - znajduje nazwę programu
Nazwa = BaseName(„Nazwacieki")
D0 D1

Funkcja:

Procedura znajduje nazwę programu w podanej ścieżce, np. „DF0:Program" - Nazwa programu będzie „Program" ale „DF1:Dane(ZbioryCode)Data" - Nazwa będzie „Data".

Wejście:

Nazwa ścieżki - zakończona zerem nazwa ścieżki.

Wyjście:

Nazwa - wskaźnik Nazwy programu.

c.d.n.

Marcin "Duddie" Dudar

JAK ZROBIĆ WŁASNE DEMO? (cz.4)

Sprites cz.2

W poprzednim artykule z tego cyklu podałem wstępne informacje dotyczące sprite'ów. Dzisiaj natomiast opiszę, jak powinna wyglądać gotowa struktura danych dla układu graficznego obsługującego sprite'y, oraz jak je animować.

I. Struktura danych.

W całości składa się z szesnastobitowych słów. Zawierają one informacje dotyczące kontroli, pozycji oraz kształtu sprite'a. Ułożone są zawsze w następującej kolejności:

```
dc.w (WORD)   Pierwsze słowo kontroli -Pionowa i
              ;pozioma pozycja startu
dc.w (WORD)   Drugie słowo kontroli   - Pionowa
              ;pozycja zakończenia
dc.w (WORD)   Dane dla kształtu i koloru linii 1
dc.w (WORD)   Dane dla kształtu i koloru linii 1
dc.w (WORD)   Dane dla kształtu i koloru linii 2
dc.w (WORD)   Dane dla kształtu i koloru linii 2
dc.w (WORD)   Dane dla kształtu i koloru linii 3
                     .
                     .
                     .
dc.w (WORD)   Dane dla kształtu i koloru linii N
dc.w (WORD)   Dane dla kształtu i koloru linii N
dc.w (WORD)   Pierwsze słowo opisujące dalsze użycie
              ;sprite'a lub 0 - koniec
dc.w (WORD)   Drugie słowo opisujące dalsze użycie
              ;sprite'a lub 0 - koniec
```

1. Pierwsze słowo kontroli.

Informuje o początkowej pozycji pionowej i poziomej objektu.

- bity 15-8 zawierają młodsze osiem bitów pozycji ;pionowej (VSTART)
- bity 7-0 zawierają starsze osiem bitów pozycji ;poziomej (HSTART)

2. Drugie słowo kontroli.

Informuje o końcowej linii wyświetlania (VSTOP), przy czym wartość VSTOP-VSTART jest wysokością sprite'a.

- bity 15-8 zawiera młodsze osiem bitów końcowej ;linii wyświetlania (VSTOP)
- bit 7 odpowiedzialny jest za specjalny tryb ;włąc zenia sprite'ów, przy którym otrzym ujemy ;objekty więcej niż 4 kolorowe.
- bity 6-3 nie są używane (ustawi na 0)
- bit 2 starszy bit VSTART
- bit 1 starszy bit VSTOP
- bit 0 młodszy bit HSTART

3. Dane dla kształtu i koloru linii.

Te dane nauczyliśmy się sporządzać w poprzednim miesiącu. Czytelnikowi należy się jednak w tym miejscu sprostowanie. Opisując ostatni etap konwersji danych bitowych użyłem błędnie operatora dc.l zamiast dc.w (szerokość sprite'a wynosi 16 bitów, a nie 32), za co bardzo przepraszam.

4. Pierwsze i drugie sowo opisujce dalsze uycie sprite'a.

Czasem może się zdażyć, że jeden objekt chcemy wykorzystać wielokrotnie. W takim przypadku te dwa ostatnie słowa stają się pierwszymi słowami kontroli i opisują dalsze użycie sprite'a. W przeciwnym wypadku ustawiamy je na zero.

Teraz, gdy struktutra jest już gotowa możemy wprowadzić sprite'a na ekran. W tym celu należy jej adres umieścić w rejestrach:

```
120 i 122 - dla sprite'a 0     130 i 132 - dla sprite'a 4
124 i 126 - dla sprite'a 1     134 i 136 - dla sprite'a 5
128 i 12A - dla sprite'a 2     138 i 13A - dla sprite'a 6
12C i 12E - dla sprite'a 3     13C i 13E - dla sprite'a 7
```

Pamiętać przy tym należy, że w/w rejestry są 18-bitowe, więc nasza struktura musi być umieszczona w pamięci CHIP. Aktualnie pozostaje nam jeszcze ustawienie najstarszego bitu rejestru 96 (DMACON) i objekt pojawi się na ekranie.

II. Kolory.

W odróżnieniu od innych komputerów (C64, ATARI 800XL) Amiga nie posiada osobnych rejestrów kolorów dla sprite'ów. Wykorzystują one te same rejestry co grafika. Może to być niewygodne w przypadku, gdy chcemy równocześnie korzystać ze sprite'ów i grafiki używając ponad 4 bit-plane'y. W związku z tym, że rejestry będą się dublowały, pojawią się kłopoty z uzyskaniem niezależnych kolorów.

A oto wykaz rejestrów odpowiadających za poszczególne sprite'y:

Numer sprit'a:	Bitowa war wzoru:	Nazwa rejestr koloru:	Adres:
0 i 1	%00 %01 %10 %11	COLOR 00 COLOR 17 COLOR 18 COLOR 19	180 1A2 1A4 1A6
2 i 3	%00 %01 %10 %11	COLOR 00 COLOR 21 COLOR 22 COLOR 23	180 1AA 1AC 1AE
4 i 5	%00 %01 %10 %11	COLOR 00 COLOR 25 COLOR 26 COLOR 27	180 1B2 1B4 1B6
6 i 7	%00 %01 %10 %11	COLOR 00 COLOR 29 COLOR 30 COLOR 31	180 1BA 1BC 1BE

III. Poruszanie.

Porusznie sprite'ów jest rozwiązaniem bardzo prostym. Wystarczy jedynie co jedną ramkę cyklicznie zmieniać jego położenie zwiększając lub zmniejszając jego współrzędne, lub pobierając je z tablic. Z uwagi na prostotę tego zagadnienia nie będę go rozwijać i proponuję zapoznać się z programem zamieszczonym w „Kąciku początkującego kodera".

Krzysztof „K.K." Kobus.

MOONBASE

W porządku, ręce do góry, wszyscy ci co grali w Sim City i pomyśleli: tak bardzo fajne, ale wolałbym coś na wyższym poziomie technicznym! Oto jest! W programie firmy Mindscape (dystrybutor gry), masz do czynienia z czymś całkowicie nowym, odmiennym na tyle, aby uzasadnić zakup tej gry, nawet jeśli posiadasz Sim City.

Baza księżycowa jest założona w surowym środowisku Księżyca. Nasa zdecydowała, że Ziemia jest zbyt przeludniona, aby mogła być samowystarczalna.

Rasa ludzka, by przeżyć musi się rozdzielić, a Księżyc jest idealny do tego celu. Nie jest zbyt odległy - używając terminologii kosmicznej - jest u naszych wrót, a odrobina pracy mogłaby go uczynić miłym miejscem do życia, spędzenia wakacji. Ale zanim Księżyc stanie się gościnny dla ludzi, trzeba odwalić trochę poważnej roboty.

Najpierw będziesz musiał dokonać przekształceń (zmian) własnościowych i wznieść kilka budynków. Miej na uwadze, że ludzie, aby przeżyć, potrzebują powietrze, żywność, wodę, ciepło i energię. Wszystko to musi być dostarczone, ponieważ nie wyprodukowano jeszcze dostatecznie długiego kabla energetycznego, mogącego połączyć Ziemię z Księżycem. Wszystko to nie jest tanie i twoja w tym głowa, jako dowódcy stacji, aby zrównoważyć budżet, a nawet wypracować dochody, zanim Nasa zdecyduje obciąć ci dotacje.

Masz 10 lat, aby zamienić skalne złomowisko w samowystarczalny biznes. Kiedy sprawy ruszą z miejsca, powinieneś zacząć zachęcać ludzi do spędzenia tutaj wakacji.

Jeśli twoje fabryki będą miały nadprodukcję, możesz ją sprzedać z powrotem Ziemi i poprawić w ten sposób swój bilans.

Grając w „Moonbase" trzeba wziąć pod uwagę trzy podstawowe elementy: załogę, energię i kontrolę cieplną. Większość struktur na twojej bazie tego wymaga i jeśli nie funkcjonują one właściwie, zaczyna dziać się niedobrze.

Jeśli dotacje z Nasa okażą się niewystarczające, co jest bardzo prawdopodobne, będziesz musiał zająć się eksportem. Możesz zbudować fabryki produkujące LLOX lub HE3, pod warunkiem, że otrzymasz za te towary na rynku godziwą cenę. Możesz także wysłać ekspedycję, poszukującą na powierzchni księżyca minerały i wodę.

Jeśli znajdziesz np. wodę nie będziesz musiał jej kupować z Ziemi i zaoszczędzisz w ten sposób pieniądze.

Aby wszystkie twoje przedsięwzięcia sprawnie działały potrzebujesz oczywiście ludzi tworzących załogę. Ludzie ci potrzebują moduły sypialne, więc musisz je zbudować, dostarczyć do nich energię i je ogrzewać. Poza tym, kto na Księżycu zechce dzielić mieszkanie z pracownikiem reaktora atomowego, który świeci w ciemnościach i spędza ci sen z powiek?

Są jeszcze inne powody bezsennych nocy, a mianowicie przeróżne katastrofy, które mogą się wydarzyć. Mogą one spowodować znaczną śmiertelność wśród załogi i w ten sposób spowolnić tempo pracy aż do całkowitego zastoju. Płomień księżycowy jest pierwszym przykładem takiej katastrofy. Załoga będzie miała osiem minut na schronienie się. Jeśli byś przewidział tą sytuację i zbudował teleskop, to straty byłyby zminimalizowane.

Wiadomo, że lądowniki zaopatrujące księżyc rozbijają się, jeśli nie ma dla nich dostatecznej ilości lądowisk. Każdy także wie, że niebezpieczne jest poleganie na siłowniach nuklearnych jako źródle energii. Jeśli musisz je używać, upewnij się, że zostały podjęte dodatkowe środki bezpieczeństwa i siłownie są umieszczone w kraterach.

System kontroli w „Moonbase" jest bardzo prosty, używasz myszy aby naprowadzić i wykryć cel, korzystasz z menu i selektorów. Każdy, kto grał w Sim City, opanuje tą metodę natychmiast.

Według mnie „Moonbase" „bije na głowę" raz i na zawsze „Sim City" i wszystkie dodatkowe dyski tej gry.

Długo czekaliśmy na taką grę, ale warto było.

Jest tyle różnych czynników, które należy spełnić, aby z powodzeniem nadzorować księżycową kolonię, że trzeba wiele nocy, aby przez to wszystko przejść. Trzeba dodać, że gracze - eksperci od Sim City poradzą sobie tylko do pewnego momentu...

Oryginalny opis gry jest przyjemnym urozmaiceniem, i dobrą rozrywką. Pierwsza część to krótkie opowiadanie, które daje ci sporo wskazówek, jak osiągnąć sukces w grze i nawet dostarcza kilka podstawowych rad. Pozostała część to obszerny przewodnik wszystkich sposobów i możliwości kontroli.

Po lekturze opisu, spróbuj jak to wszystko działa na powierzchni księżyca!

Ta gra TO AS!

Gra zawiera bardzo mało muzyki (tylko w części tytułowej), później są za to dobre efekty dźwiękowe. Wszystko to daje zdumiewającą atmosferę stworzoną przez wielką grę.

Jest to jedna z najbardziej „wciągających" gier. Zasięg jej jest prawie nieograniczony. Wspólnie z „Eye of the Beholder" jedna z najlepszych gier roku. **Klasyka.**

Na podstawie „Moonbase", Amiga Computing, nr 08/91r.

opracował H.S.

PUBLIC DOMAIN PACK

PUBLIC DOMAIN PACK C-64

Kwiecień

Strona A

- Digi - Organizer - program do tworzenia muzyki z użyciem digitalizacji dźwięku.

Strona B

- „ONE YEAR - RADIUS" - mega demo grupy RADIUS. Bardzo ładna grafika.

Maj

Strona A

- CRUEL SOLIDERS - demo
- DESTINATION'91 - demo
- SUCKER DJ! - demo (digi mix)
- MUSIC SEARCHER - do wycinania ilustracji muzycznych z programów

Strona B

- MEGA DEMO „INFOSYSTEM 91"

Czerwiec

Strona A

- FONTEDITOR
- SINDATA EDITOR
- COLOR EDITOR
- DISK - NOTER
- GWIAZDY - demo graficzne
- FILGRAEPH 2.2/BML
- NOTE TO FLI V 2.2
- AFLI - EDITOR V 1.2
- RESET - MON,8,1
- TURBO - ASS 5
- ...HIGHLIFE #5
- AXEL NEWS #1
- DISK NOTKA/PADUA

Strona B

- PSC - MAG #9'06/91
- CONSPIRE/OREGON - demo
- CONTACT DEMO/ORE
- SHOWPIX

Lipiec

Strona A

- Mega demo „MY, OH MY!" grupy LIGHT

Strona B

- Game Music Composer - edytor muzyczny grupy GRAFFITY Węgier.

Sierpień

Strona A

- MegaDemo „Unnamed" grupy CAMELOT
- Sound Killer - edytor muzyczny grupy TOPAZ
- AFLI - Editor - edytor graficzny techniki A-FLI
- Disk-Dos obsługa komend stacji dysków
- Noter v2.2 grupy TOPAZ
- IFFL - Squeezer kompresor dyskowy
- Dirmaster+ - edytor do dyskietek
- Super Copy - DOS szybki program kopiujący do zbiorów

Strona B

- Mega Demo fińskiej grupy TOPAZ - „Graveyard Blues"

PUBLIC DOMAIN PACK AMIGA

Maj

- VIRUS X 5.0
- VIRUS TERMINATOR
- PARADOX - demo
- STORMCHILD - demo
- Moduły muzyczne:
  - MIAMI VOICE
  - ANTI ATARI SONG

Czerwiec

- POWER BOOT - własne menu dysku
- DISK CODING SYSTEM - program do zabezpieczania dysków
- Konwerter IFF - ANSI
- AUER NATION - demo
- Moduły muzyczne
- DOCS - opis gry ELWIRA
- LAMER DEFENCE - do wykrywania i niszczenia wirusów
- REWENG GO OF THE LAMER - grafika w trybie D_HAM

Lipiec

- Sanity - demo
- Amiga - Tanx (1Mb) - gra
- Little Beau (1MB) - gra
- There is A Light/Tonid - modules

Sierpień

- Real 3D - demo nowego programu do raytraceing'u
- Moduł Muzyczny XTC STCREO

Wrzesień

- MODUŁY MUZYCZNE dla programu TFMX:
  > R - TYPE
  > THE HOUSE OF TECHNO
- VIRUS EXPERT v181- + 143
- BOOT BLOCK'I
  > BOOTX v 3.80
  > IMPLODER v 4.0

Spis treści zestawów z poprzednich miesięcy - patrz wcześniejsze numery naszego pisma.

Zestawy „64 plus 4 PUBLIC DOMAIN PACK" można zamawiać wpłacając na konto: Bank PKO SA Oddział w Bydgoszczy konto nr:5.09011-400522.7-136-11-111.0 następujących kwot: 20.000zł za pojedyńczy zestaw dla C-64, 25.000zł za zestaw dla AMIGI. Kwoty te obejmują koszt dyskietki, koszty kopiowania, opakowania i przesyłki pocztowej. Blankiety wpłat powinny być **CZYTELNIE** wypełnione i zawierać: **imię i nazwisko, dokładny adres zamawiającego, skrót „PDP-64"**(jeśli zamawiamy zestaw dla C-64) lub **„PDP-A"**(zestaw dla Amigi) - dane te prosimy umieszczać na **wszystkich** odcinkach dowodu wpłaty. W prenumeracie zestawy kosztują: PDP-64 - 18.000zł (12 numerów 216tys zł), PDP-A - 22.000zł (12 numerów 264tys zł). Prenumeratę można zawrzeć w dowolnym terminie na okres od 3 do 12 miesięcy (do końca roku kalendarzowego). Prenumerata może obejmować miesiące od początku roku - tzn. zamawiając całoroczną prenumeratę np. w październiku, w pierwszej przesyłce otrzymacie wszystkie poprzednie zestawy.

ZAMÓW NIE ZWLEKAJ!